HARDWARE IMPLEMENTATION OF SPIKING NEURAL NETWORKS

Dissertation submitted to
Visvesvaraya National Institute of Technology, Nagpur
In partial fulfillment of the requirements for the award of the degree

Master of Technology
in
VLSI Design

by

Shaik Thotti Mahammad Ali
(MT23MVD001)

Under the guidance of

Dr. Sanjay B. Dhok

Center for VLSI and Nanotechnology
Visvesvaraya National Institute of Technology
Nagpur 440010 (India)

June 2025

HARDWARE IMPLEMENTATION OF SPIKING NEURAL NETWORKS

Dissertation submitted to
Visvesvaraya National Institute of Technology, Nagpur
In partial fulfillment of the requirements for the award of the degree

Master of Technology
in
VLSI Design

by

Shaik Thotti Mahammad Ali
(MT23MVD001)

Under the guidance of

Dr. Sanjay B. Dhok

Center for VLSI and Nanotechnology
Visvesvaraya National Institute of Technology
Nagpur 440010 (India)

June 2025

©Visvesvaraya National Institute of Technology, Nagpur 2025

Center for VLSI and Nanotechnology
Visvesvaraya National Institute of Technology
Nagpur

DECLARATION

I, **Shaik Thotti Mahammad Ali**, here by declare that this dissertation titled **"Hardware implementation of Spiking Neural Networks"** is carried out by me in the Center for VLSI and Nanotechnology of Visvesvaraya National Institute of Technology, Nagpur. This work is original and has not been submitted earlier whole or in part for the award of any degree/diploma at this or any other Institute/University

Shaik Thotti Mahammad Ali
Date:

CERTIFICATE

This is to certify that the dissertation titled **"Hardware implementation of Spiking Neural Networks"** is submitted by **Shaik Thotti Mahammad Ali (MT23MVD001)** fulfills the partial requirements for the Master of Technology degree in VLSI Design at VNIT Nagpur. The work is thorough complete and suitable for final evaluation.

Dr. S.B. Dhok
Head, Center for VLSI and Nanotechnology
VNIT, Nagpur

Dr. S.B. Dhok
Center for VLSI and Nanotechnology
VNIT, Nagpur

Center for VLSI and Nanotechnology

Visvesvaraya National Institute of Technology, Nagpur

NON-PLAGIARISM CERTIFICATE

Certified that M. Tech Dissertation titled "**Hardware implementation of Spiking Neural Networks"** is submitted by **Shaik Thotti Mahammad Ali (MT23MVD001)** has been checked for plagiarism using TURNITIN software and the overlap is found within prescribed limits. A summary of the report is given below,

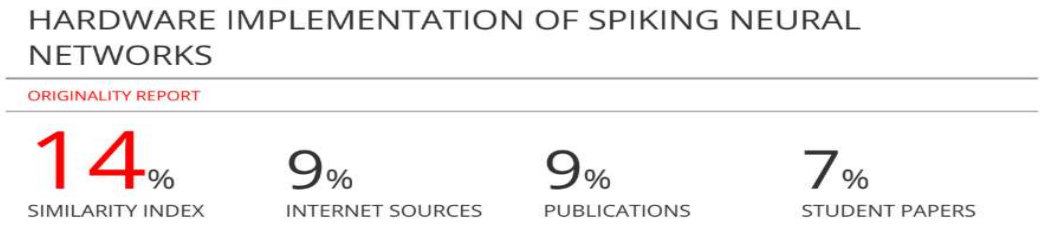

Fig 0 : Plagiarism Certificate from TURNITIN Software

.

S.T. Mahammad Ali

Date:

Dr. Sanjay B. Dhok
Supervisor

ACKNOWLEDGMENT

Despite my name being the only one on the cover, many others helped develop this research. I am sincere in my gratitude to everyone who made my project work possible.

I would like to express my sincere regards to **Prof. P. L. Patel**, Director of the Institute; **Prof. V. R. Kalamkar**, Dean (Academics); and **Prof. R. V. Taiwade,** Associate Dean (Academics – PG/PhD), **Prof. Sanjay B. Dhok**, Head of the Department, Center for VLSI and Nanotechnology; for their unwavering support and commitment towards fostering an academic environment that empowers students to excel. Their leadership and support have been instrumental in providing the resources and encouragement needed to complete this project work.

My deepest gratitude is to my advisor, **Dr. Sanjay B. Dhok**. I consider myself extremely fortunate to have a mentor like him, who gave me the freedom to conduct independent research while giving me the guidance I needed to get back on track when I made mistakes. I had some enlightening conversations and constructive criticism during my project that got me to reflect and helped me focus. His understanding and support allowed me to get through several challenging circumstances and finish my project.

Additionally, I would like to thank to **Dr. R.M.Patrikar , Dr. R.B.Deshmukh, Dr. P.Vasu, Dr. Nikhil Deep Gupta,** and **Dr. G.C Patil** for their constant advice that improved my project work. I appreciate my classmates support in keeping me motivated while I worked on my project.

Above all, none of these achievements would have been attainable without the love and patience of my family. I want to extend my sincere appreciation to my family.

Shaik Thotti Mahammad Ali

ABSTRACT

Spiking Neural Network (SNN) is a biologically inspired AI method that uses discrete spikes and temporal coding to handle information like the human brain. SNNs are different from traditional neural networks in that they work based on the timing of spikes, thus being very energy efficient and ideally suited for event-driven systems. Hardware realization of SNNs focuses on capitalizing on these characteristics by developing neuromorphic systems that can provide real-time, low-power computation. Spiking neurons, synapses, and learning, such as spike-timing-dependent plasticity (STDP), are facilitated by a variety of technologies, such as FPGAs, ASICs, and emerging devices like memristors, that are utilized in a wide variety of applications.

Although promising, hardware implementation of SNNs has a number of challenges, including the complexity of the dynamics of neurons, memory size issues, and the unavailability of effective training algorithms that are compatible with spiking behaviour. Further, bringing SNNs into conventional digital infrastructure involves novel design paradigms and development toolsets. However, continuous innovation in circuit design, learning models, and co-design of hardware and software is slowly bridging these challenges. As a consequence, SNN hardware is increasingly becoming feasible for applications like robotics, sensory data processing, and edge AI. With ongoing research and innovation, SNN hardware has the potential to be at the centre of the future intelligent, adaptive, and low-power computing systems. The proposed architecture for SNN in this thesis provides low power of 0.28W and used for real time image classification application which consumes power of 1.28W.

Keywords: Spiking Neural Network, Field Programmable Gate Array (FPGA), Spike timing dependant plasticity (STDP).

LIST OF FIGURES

Fig 2.1 CNN Architecture 8

Fig 2.2 SNN Architecture 9

Fig 2.3 a) Artificial Intelligence 11

Fig 2.3 b) Hardware platforms comparison 11

Fig 2.4 Overall architecture of SNPU 12

Fig 2.5 Architecture of FPGA-NHAP Platform 13

Fig 2.6 Architecture of SNN Accelerator 14

Fig 2.7 Architecture of a single core in ESSA 15

Fig 3.1 Neuron Model 20

Fig 3.2 SNN Methodology 21

Fig 3.3 LIF Neuron Working principle.....23

Fig 3.4 STDP Graph for weights update.....27

Fig 3.5 Rate encoding technique circuit.....28

Fig 3.6 Waveform showing Image pixel encoding.....29

Fig 3.7. Architecture on FPGA.....29

Fig 3.8 SNN Accelerator Architecture.....31

Fig 3.9 Internal Structure of Neuron Core.....33

Fig 4.1 Basys 3 FPGA Board.....36

Fig 4.2 Architecture of Xilinx Artix 7 FPGA.....37

Fig 4.3 AXI protocol.....38

Fig 5.1 Result of Software simulation.....41

Fig 5.2 Rate encoder circuit Schematic view.....41

Fig 5.3 Simulation of Rate encoding technique.....42

Fig 5.4 Integrate and Fire Neuron Model Schematic view.....42

Fig 5.5 LIF Neuron Model.....43

Fig 5.6 Schematic view of LIF Neuron.....43

Fig 5.7 Simulation result of LIF Neuron.....44

Fig 5.8 Log file of above simulation result.....44

Fig 5.9 Synthesis result of LIF Neuron.....45

Fig 5.10 SNN Architecture implemented on FPGA.....45

Fig 5.11 Graph showing % of components utilized.....46

Fig 5.12 Power utilization Report.....47

Fig 5.13 Summary of Power consumption.....47

Fig 5.14 Timing Report.....47

Fig 5.15 Output of the input test image1 on Basys3..... 48

Fig 5.16 Output of the input test image2 on Basys3..... 48

Fig 5.17 Output of the input test image3 on Basys3..... 48

Fig 5.18 Output of the input test image4 on Basys3..... 49

Fig 5.19 Output of the input test image5 on Basys3..... 49

Fig 5.20 Power report of image classification.....49

LIST OF TABLES

Table 3.1 Parameters for controlling spiking neuron.....24

Table 3.2 Comparison between Different types of spiking Neuron Models.....26

Table 5.1 Utilized components list.....46

ABBREVIATIONS

AI	Artificial Intelligence
ANN	Artificial Neural Networks
CNN	Convolutional Neural Networks
SNN	Spiking Neural Networks
STDP	Spike Time Dependant Plasticity
ASIC	Application Specific Integrated Circuit
FPGA	Field Programmable Gate Array
IF	Integrate and Fire
LIF	Leaky Integrate and Fire
IZH	Izivikeich
ML	Machine Learning
TPU	Tensor Processing Unit
GPU	Graphic Processing Unit
AXI	Advanced eXtensible Interface
AHB	Advanced High-Performance Bus
DDR	Double Data Rate
DMA	Direct Memory Access
NHAP	Neuromorphic Hardware Acceleration Platform
LUT	Look- Up Table
CLB	Configurable Logic Block
BRAM	Block Random Access Memory
IC	Integrated Circuit

TABLE OF CONTENTS:

ABSTRACT.....i
LIST OF FIGURES.....ii
LIST OF TABLES.....iv
ABBREVIATIONS.....v
Chapter 1: INTRODUCTION

1.1 Background.....1
1.2 Motivation.....3
1.3 Objective.....4
1.4 Report Organization.....5

Chapter 2: LITERATURE REVIEW

2.1 Machine Learning.....7
2.2 Neural Networks.....8
2.2.1 Convolutional Neural Network Architecture 8
2.2.2 Spiking Neural Network Architecture.....9
2.3 FPGA vs ASIC implementation.....16

Chapter 3: METHODOLOGY

3.1 Neuron 20
3.1.1 Spiking Neural Network.....21
3.2 Neuron Models.....22
3.2.1 Integrate and Fire Model.....22
3.2.2 Leaky Integrate and Fire Model.....23
3.2.3 Izhikevich Model.....24
3.2.4 Hodgkin- Huxley Model.....25
3.3 Spike Time Dependent Plasticity.....26
3.4 Image Encoding Techniques.....28
3.4.1 Rate Encoding.....28
3.4.2 Temporal Encoding.....28
3.5 Architecture.....29
3.5.1 SNN Accelerator.....31

Chapter 4 : HARDWARE IMPLEMENTATION

4.1 Hardware tools..... 34

4.1.1 FPGA.....34

4.1.2 Types of Boards..... 35

4.1.3 Basys 3 FPGA Board..... 36

4.1.4 AXI Protocol..... 37

4.2 Vivado Software..... 38

Chapter 5: RESULTS

5.1 Software Results.....41

5.2 Image Encoding technique results..... 41

5.3 Neuron Model simulations.....42

5.3.1 IF Model.....42

5.3.2 LIF Model.....43

5.4 SNN Architecture.....45

5.5 Reports.....46

5.5.1 Utilization Report.....46

5.5.2 Power Report.....47

5.5.3 Timing Report.....47

5.6 Image Clasification…….....48

5.6.1 Power Report of image classification.....49

Chapter 6: CONCLUSION AND FUTURE SCOPE

6.1 Conclusion.....51

6.2 Future Scope.... 51

REFERENCES.....52

1

INTRODUCTION

1.1 Background

Neuromorphic computing is a fast-developing discipline aimed at creating computing systems inspired by structure and operation of human brain. It tries to overcome traditional von Neumann architecture limitations by emulating the parallel, event-driven, and highly energy-efficient properties of biological neural systems. This paradigm is especially significant in the age of artificial intelligence (AI), when the need for real-time computation and power-efficient computation is greater than ever. Two of the most powerful models applied in neuromorphic computing are Convolutional Neural Network (CNN) and Spiking Neural Network (SNN), each of which plays a major role in this field.

Foundations of Neuromorphic Computing

The philosophy of neuromorphic computing was pioneered by Carver Mead during the late 1980s with the aim of designing hardware systems that mimic the physical and functional behavior of neural circuits. The systems employ analog or digital circuits to simulate neurons and synapses so that low-power, high-speed computing platforms can be built. Neuromorphic hardware tends to be event-driven and asynchronous in nature, which is starkly different from the traditional clock-driven, sequential computers that process information.

The brain is able to process sophisticated cognitive functions — including vision, learning, and motor control — with remarkable efficiency, using approximately 20 watts of electricity. Neuromorphic systems aim to match this efficiency by embracing brain-inspired models such as CNNs and SNNs, each with their own benefits.

Convolutional Neural Networks (CNNs)

CNNs are one of the deep learning models best suited for image and pattern recognition. Motivated by the visual cortex, CNNs use convolutional filter layers to identify hierarchical data features. Although CNNs are not spiking or event-driven in nature like SNNs, CNNs have been effectively converted to suit neuromorphic computing, particularly on edge AI devices where power consumption is paramount.

Neuromorphic hardware implementations of CNNs typically include hardware accelerators that optimize the inference process — the step during which the learned network predicts. Dedicated chips such as Intel's Loihi and IBM's TrueNorth have investigated techniques for executing reduced or quantized CNNs to save energy and decrease latency. CNNs can also

be used as a preprocessing step in hybrid setups, with features obtained through CNNs processed by an SNN for decision-making.

Spiking Neural Networks (SNNs)

SNNs are regarded as the third generation of neural networks and are biologically closer than CNNs or standard ANNs. In contrast to CNNs that handle data in continuous, real-valued signals, SNNs utilize discrete spikes — time events similar to the way actual neurons convey information. A spiking neuron sums incoming spikes, and upon achieving a certain threshold, it releases its own spike. Models like the Leaky Integrate-and-Fire (LIF) neuron are typical in mimicking this.

SNNs are naturally compatible with neuromorphic hardware owing to their sparse and event-driven nature. They only draw power when events (spikes) take place, and hence they are perfect for real-time and power-constrained systems like robotics, wearable systems, and brain-machine interfaces. SNNs are difficult to train because spikes are non-differentiable, and hence backpropagation-based techniques cannot be used in full force. New training strategies, such as surrogate gradient learning and biologically motivated learning rules such as spike-timing-dependent plasticity (STDP), are working to overcome this limitation.

Hybrid Approaches and Applications

Hybrid neuromorphic systems combining CNNs and SNNs are a promising avenue of development. CNNs might be applied for high-precision feature extraction, and SNNs would be able to process event-driven information and detect temporal patterns. These architectures are particularly beneficial for processing spatiotemporal data such as video, sensor data, and audio, where timing is of essence.

Neuromorphic computing is becoming increasingly applicable across areas like self-driving cars, smart monitoring, speech understanding, and health diagnostics. By making the most of both CNNs and SNNs, neuromorphic systems are on course to provide strong and efficient AI capability directly on devices, without the need for cloud processing.

1.2 Motivation

The motivation to create specialized hardware for Spiking Neural Networks (SNNs) is due to the drive for efficient, brain-mimicking computation that can surpass the constraints of traditional AI systems. Traditional deep learning techniques like those using artificial neural networks (ANNs) are computationally intensive and consume a lot of energy, and thus are not applicable in low-power or real-time environments. Conversely, SNNs provide a biologically realistic model of neural processing that employs sparse event-driven communication via spikes, similar to the brain.

Energy efficiency is one of the main drives for SNN hardware. As neurons in an SNN will only spike (or become active) when a threshold is crossed, computation is only invoked where necessary. This results in sparse activity and drastically lower power utilization, particularly compared to dense matrix operations employed by ANNs and CNNs. This aspect is especially beneficial for edge computing scenarios like wearables, IoT sensors, and autonomous systems, where power budgets are constrained.

Low-latency, real-time processing is another major driving force. SNNs inherently deal with temporal and sequential data, e.g., audio or event-based vision data, and are thus particularly suitable for time-critical applications such as robotics, gesture recognition, and brain-computer interfaces. Specialized hardware can take advantage of the parallelism and asynchrony of SNNs to provide immediate responses without the need for bulk batch processing.

In addition, SNN hardware facilitates the application of neuromorphic learning mechanisms, including spike-timing-dependent plasticity (STDP), which makes on-chip learning more adaptive and scalable. This facilitates lifelong learning and dynamic adaptation in dynamically changing environments — features difficult for conventional AI models to accomplish.

.

1.3 Objective

The objective of implementing Spiking Neural Network (SNN) hardware on Field Programmable Gate Arrays (FPGAs) for image recognition is to develop an energy-efficient, low-latency, and real-time processing system that mimics the neural behavior of the human brain while achieving competitive accuracy in visual recognition tasks.

Key goals include:

1. **Energy Efficiency**:
   Design and deploy a hardware-efficient SNN architecture on FPGA that leverages the sparse, event-driven nature of spiking neurons. FPGAs are well-suited for custom low-power designs, making them ideal for implementing neuromorphic models with minimal energy consumption compared to traditional CPUs/GPUs.
2. **Real-Time Processing**:
   Utilize the inherent parallelism and reconfigurability of FPGAs to implement SNNs capable of processing image data in real time. This is crucial for applications such as robotics, surveillance, and edge AI where fast decision-making is essential.
3. **Biological Plausibility and Neuromorphic Computing**:
   Emulate brain-like computation by incorporating neuron models such as Leaky Integrate and Fire (LIF) and applying spike-based encoding techniques (e.g., rate coding or temporal coding) for image data. This supports the development of hardware systems inspired by cognitive processes.
4. **Compact and Scalable Architecture**:
   Create an architecture that is resource-efficient on the FPGA platform, allowing for scalability in network size and the ability to handle higher-resolution images or more complex classification tasks.

1.4 Theses Organization

The theses is organized into 6 chapters as discussed below.

Chapter 1

Discusses the introduction of the project, motivation for doing this research work, project objectives, and thesis organization.

Chapter 2

Provides a basic understanding of the Machine Learning and different types of architectures for implementing Neural Networks. The implementation of the architecture for spiking neural networks on FPGA is also covered. The challenges associated with implementation are also discussed.

Chapter 3

Presents the methodology of Spiking Neural Networks with its types and comparision between them. Also presents the image encoding techniques used to convert pixel into spikes for processing.

Chapter 4

This chapter shows the hardware implementation and explains about how FPGA works and how Vivado software is used for simulations of the design netlist.

Chapter 5

Result obtained from the Vivado software and implementation on hardware.

Chapter 6

Project conclusion and its future scope.

2

LITERATURE REVIEW

2.1 MACHINE LEARNING

Machine Learning (ML)is a subset of artificial intelligence (AI) that aims to create systems capable of learning from data and enhancing their performance with the passage of time without being directly programmed. Rather than programming code with a set of instructions, programmers provide input data to algorithms that go on to determine patterns, make judgments, or forecast outcomes.

Machine Learning is classified into three categories:

- supervised learning,
- unsupervised learning,
- reinforcement learning.

In supervised learning, algorithm is learned over labeled data, i.e., the input is accompanied by related output values (e.g., house price prediction based on features such as size and location). Unsupervised learning is applied to unlabeled data and attempts to extract unknown structures or patterns from it (e.g., customer segmentation). Reinforcement learning is the process of training an agent to make a sequence of decisions based on rewarding good behavior and penalizing bad behavior (e.g., robotics).

Machine learning is extensively applied in real-world problems like recommendation systems (e.g., Netflix or Amazon), image and speech recognition, natural language processing (e.g., chatbots), fraud detection, self-driving cars, and medical diagnosis.

The success of an ML model also relies substantially on the quantity and quality of the data, the selection of algorithm, and the capacity to tune model parameters. Common algorithms are decision trees, support vector machines, neural networks, and ensemble techniques such as random forests and gradient boosting.

Recent breakthroughs in deep learning, which is a branch of ML employing multi-layered neural networks, have dramatically enhanced performance in image classification, speech generation, and translation.

While its capability is immense, ML also has drawbacks, such as data bias, non-interpretable outputs, and requirements for enormous amounts of computational resources. Nevertheless, it remains a revolutionary technology that is transforming sectors and rewriting the rules of how machines engage with the world.

2.2 Neural Network Architecture

2.2.1 Convolutional Neural Network Architecture

The architecture of the Convolutional Neural Network (CNN) is well-structured and follows a standard deep learning pipeline for image classification. The pipeline begins with feature extraction, where multiple layers of convolution, each followed by a ReLU activation and a pooling layer, try to learn and extract features increasingly from the input image. Convolution operations are utilized to detect local patterns such as edges and textures, with ReLU bringing in a non-linear element, allowing the network to learn higher-order functions. Pooling layers, often utilizing max-pooling, downsample spatial dimensions, thus improving computational efficiency while encouraging translation invariance.

In the classification stage, the feature maps are transformed by a flattening layer into a one-dimensional vector. The vector is input into fully connected (dense) layers, which perform complex reasoning on the learned features. The final layer employs a SoftMax activation function to output a probabilistic class distribution over different classes. This allows the model to determine the most likely label for the image, i.e., label it as a zebra with maximum confidence. The combination of dense and convolutional layers offers a strong architecture for processing and labeling complex visual information

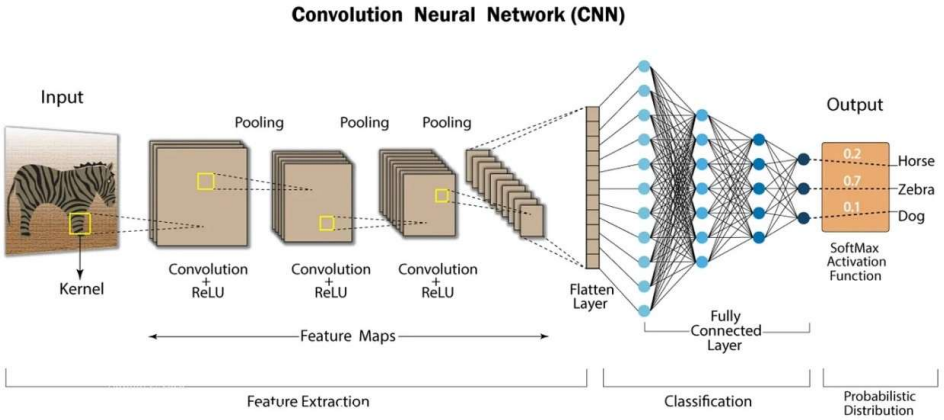

Fig 2.1 CNN Architecture [21]

The picture shows the structure of a Convolutional Neural Network (CNN), one of the most common deep learning structures for image classification. It starts with an input image—our case in point is a zebra—then a sequence of convolutional layers with ReLU (Rectified Linear Unit) activations. The convolutional layers use optimized filters (kernels) that are tuned to extract low-level features like edges and texture. Following each convolutional layer, there are pooling operations to reduce the spatial dimensions of the feature maps, thus reducing computational intensity and enhancing the capacity of the model to detect features in spite of their spatial location in the image.

Following the feature extraction process, the output data is flattened into a representation of a single vector and is passed to fully connected layers. These are a typical neural network classifier, using the extracted features and calculating the most likely class. The output layer then uses the SoftMax activation function, normalizing the raw scores to the probability for each category. In this example, the network is predicting that the image is a zebra with 70% confidence, to horse with 20%, and to dog with 10%. This structure allows CNNs to excel in tasks involving complex visual data

2.2.2 Spiking Neural Network Architecture

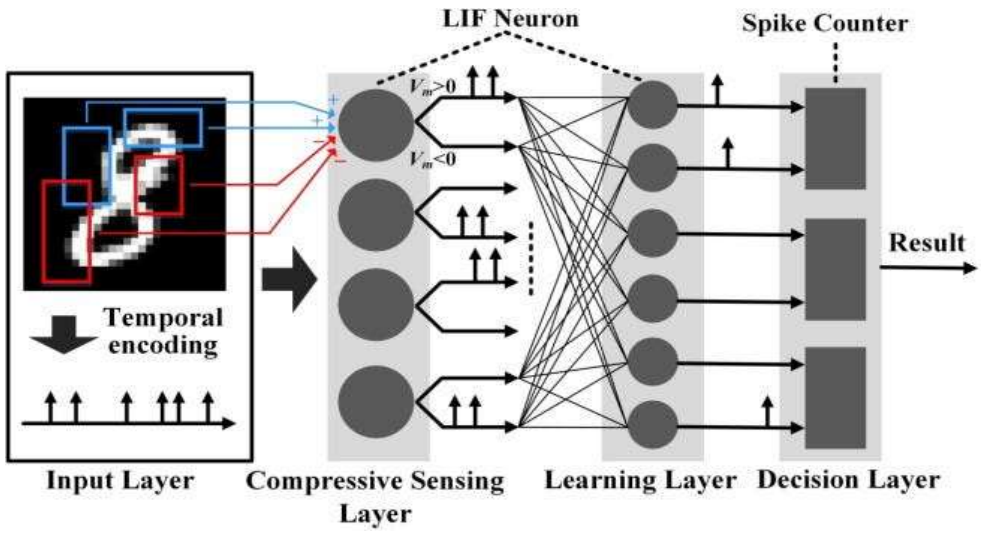

Fig 2.2 SNN Architecture [22]

The figure depicts a high-level schematic diagram of a Spiking Neural Network (SNN) architecture operating on bio-inspired mechanisms. In the Input Layer, the input image—assumed to be a digit—is temporally encoded—converting pixel values to spike trains over time. The mapping is done on the mechanism of information transmission in biological neurons, not on a continuous signal but spikes. The resulting spiking activities are now fed into the Compressive Sensing Layer, which compresses the input data in terms of dimension. This layer is required for efficient processing and is derived on the basis of the compression of sensory information in the brain before interpretation. The arrows among the layers show the flow of spike events, and the symbol $V_m > 0$ and $V_m < 0$ shows the membrane potential of neurons, thus controlling their probability of sending spikes.

Spikes are passed to the Learning Layer, which consists of Leaky Integrate-and-Fire (LIF) neurons—biologically inspired models that sum the input until a threshold is reached, at which time they fire a spike. They are responsible for learning in Spiking Neural Networks (SNNs) by modifying their synaptic weights according to spike-timing-dependent plasticity (STDP) or additional learning rules. The spiking is passed to the Decision Layer, where output neurons receive the cumulative spikes. A Spike Counter tallies these spikes over an interval, and the spiking neuron with the largest spike count is used to calculate the Result. This is similar to classification tasks, where the most dominant response is used to determine the predicted class. This architectural design is apt for low-power, event-driven applications like neuromorphic systems and edge computing and provides both energy efficiency and biologically inspired learning processes.

Capra et al. [1] provide a comprehensive survey of the latest hardware and software techniques used to accelerate Deep Neural Networks (DNNs). Over the past decade, deep neural networks (DNNs) have revolutionized fields such as healthcare, finance, robotics, and autonomous systems. However, their complex architectures demand immense computational power, making traditional hardware like CPUs inadequate for real-time, energy-efficient deployment. The paper surveys the evolution of both hardware (CPU, GPU, FPGA, ASIC) and software strategies designed to accelerate DNN performance. It discusses how FPGAs and ASICs, in particular, provide scalable, low-power alternatives by leveraging parallelism

and data reuse. The survey also highlights various neural models, such as convolutional and spiking neural networks, and explains how their processing needs influence hardware design and optimization choices.

To meet growing demands in AI applications, the paper explores innovative dataflows and co-design techniques that improve memory access, computational efficiency, and energy consumption. For instance, the authors emphasize the importance of spatial architectures in hardware accelerators like Google's TPU and Eyeriss, which efficiently manage weight, input, and output reuse. Additionally, spiking neural networks (SNNs) are gaining attention for their brain-inspired efficiency and low power use. This comprehensive survey serves as a guide for researchers and engineers aiming to design optimized, future-ready AI hardware that balances performance, power, and scalability.

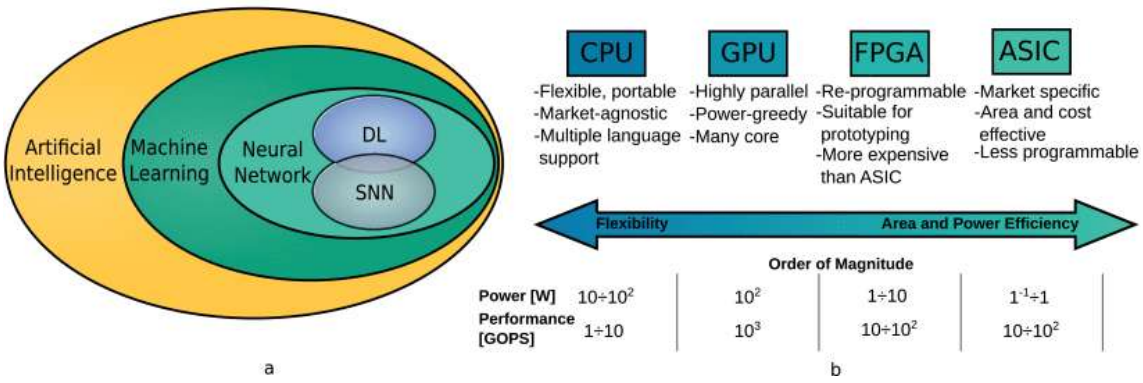

Fig 2.3 a)Artificial Intelligence b) Hardware platforms comparision [1]

Spiking Neural Network became a popular method for AI that uses less energy, especially for always-on systems like face recognition. By employing event-driven computation, the SNPU (Spiking Neural Network Processing Unit) architecture put forth by **Kim et al. [2]** suggests a spike-domain DNN processor that significantly lowers power consumption. SNNs take advantage of natural data sparsity, in contrast to traditional DNNs, which depend on dense, fixed computation. In order to reduce power-hungry operations and address inefficiencies in

previous systems that had trouble with real-world input conditions, the paper presents two-step spike encoding, also known as neural and computing encoding.

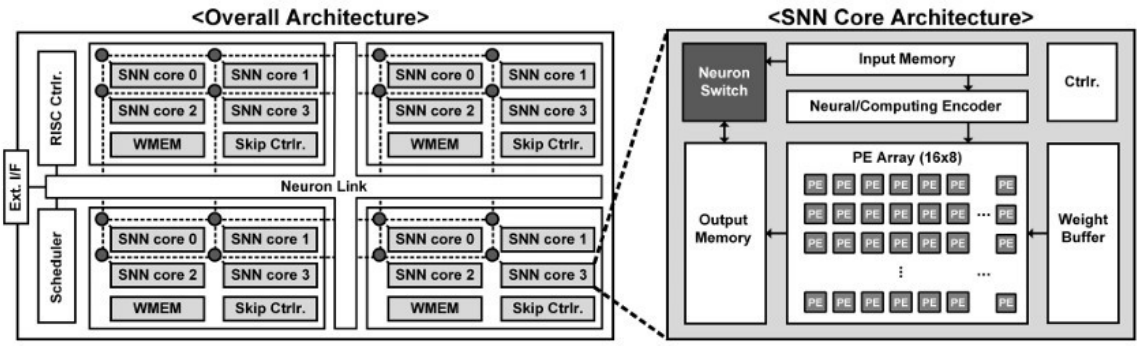

Fig. 2.4 Overall architecture of SNPU [2]

SNPU's layered optimization approach is what makes it unique. The neural encoder conserves power spent on encoding by up to 99.4% through the removal of the need for energy-hungry pseudo-random generators. On the other hand, the computing encoder conserves the number of operations by 83.3% through spike-train decomposition coupled with multi-level encoding methodologies. These developments enable SNPU to achieve 80.3% lower power draw during inference by substituting simpler Shift-and-Accumulate (SAC) logic for conventional MAC operations. When combined, these improvements allow SNPU to function well even when input and weight data are sparse, which greatly increases its adaptability to a wide range of AI tasks.

Additionally, SNPU directly addresses architectural inefficiencies. It increases PE utilization by 23.1% by managing variable time-window sizes across network layers through its Neuron Link and dynamic workload allocation. In order to manage both high and low sparsity scenarios with the least amount of overhead, the omni-sparsity handling unit deftly blends coarse- and finegrained logic. No matter the sparsity pattern, SNPU maintains state-of-the-art efficiency thanks to its modular, flexible design, which is an amazing accomplishment for practical AI applications.

Liu et al. [3] introduce **FPGA-NHAP(Neuromorphic Hardware Acceleration Platforms)**, a scalable and power-efficient neuromorphic hardware platform designed to accelerate Spiking Neural Network (SNN) inference on FPGAs.

Neuromorphic computing, inspired by the brain's efficiency, has gained attention for tasks like image recognition. Artificial neural networks, though powerful, struggle with power consumption and biological accuracy. Spiking Neural Networks offer a more brain-like approach by using timed spikes for communication. However, SNNs are complex to implement on standard CPUs and GPUs, leading researchers to explore FPGAs for their parallelism and low energy use. Existing hardware platforms often support only one type of neuron model and lack efficient algorithms, limiting their flexibility and real-world application.

To address these issues, the authors proposed FPGA-based platform capable of simulating both LIF and Izhikevich models. It includes a custom-designed neuron computing unit, an efficient event-driven update algorithm, and a hardware accelerator built with RISC-V integration. Tested on MNIST(Modified National Institute of Standards and Technology) and Fashion-MNIST datasets, the platform achieves high accuracy and frame rates while maintaining very low power consumption. Compared to prior systems, FPGA-NHAP improves scalability, speed, and power efficiency, making it a promising solution for real-time SNN deployment in edge AI and embedded systems

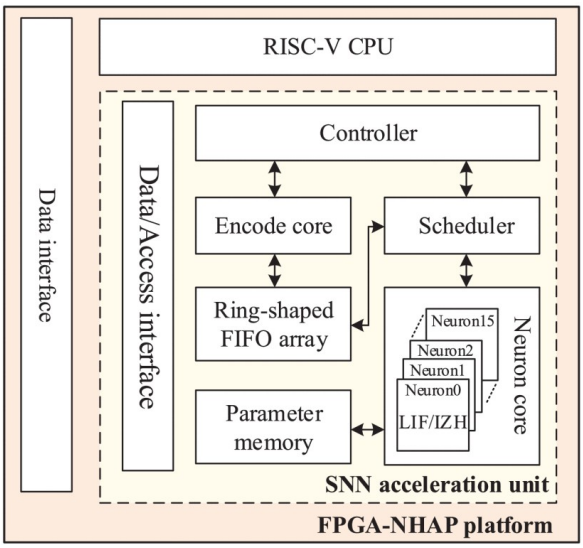

Fig 2.5 Architecture of FPGA-NHAP Platform [3]

Cheng et al. [4] suggest an event-driven, high-efficiency SNN accelerator that is tailored for handling Dynamic Vision Sensor (DVS) data. While DVS produces sparse, asynchronous events, traditional vision sensors record dense, synchronous data. The difficulty of matching spiking hardware to such inputs is discussed in this paper. Large-scale SNN deployment with structured sparsity is made possible by the design, which supports up to 1024 neurons and 1 million synapses, lowering resource consumption and increasing energy efficiency.

The architecture ensures balanced computation loads across processing elements (PEs) by imposing structured sparsity through the use of modular arithmetic. This makes hardware routing easier and lowers memory overhead. Each PE has a four-stage pipeline that speeds up neuron updates even more. Address-Event Representation (AER) data from DVS is processed directly by the accelerator, doing away with preprocessing and enabling real-time inference performance. Backpropagation through time, which is used in the training process, enables precise spatiotemporal pattern classification in DVS data streams.

At 50% sparsity, the system achieves 96% accuracy on N-MNIST and 79% accuracy on DVS-Gesture datasets when tested on a Xilinx ZCU104 FPGA.Its speed, accuracy, and low power consumption make it ideal for edge AI applications where latency and power limitations are crucial, such as robotic vision and gesture recognition.

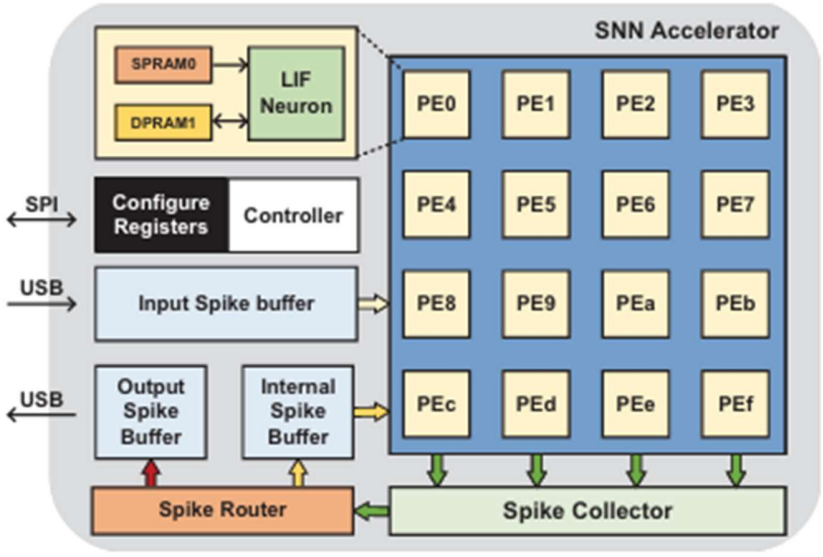

Fig. 2.6 Architecture of SNN Accelerator [5]

ESSA, an Effective Sparse SNN Accelerator designed to take advantage of temporal and spatial sparsity in spiking neural networks, is presented by **Kuang et al.** [6] Self-adaptive spike compression and compressed sparse row (CSR) formatted weights are supported by ESSA, which is made for FPGA implementation. By addressing the shortcomings of dense neuromorphic architectures in managing sparse activity, these optimizations greatly increase throughput and decrease memory traffic.

One noteworthy aspect of ESSA is its mechanism of combinable dendrites, which overcomes the limitations of traditional crossbars and permits flexible fan-in/fan-out configurations. Its 4x4 array's 16 cores reduce superfluous processing by compressing sparse input spikes prior to computation. A deadlock-free packet routing algorithm is also used by the system to support multicast communication, reducing on-chip communication latency.

Synthesized on a Xilinx Kintex Ultrascale FPGA, ESSA achieves 253.1 GSOP/s and 32.1 GSOP/W at 75% sparsity and operates at 140 MHz. The four-layer fully connected SNN model incurs 2.6 μs per timestep and 14.6 μJ. ESSA provides unmatched flexibility, scalability, and energy efficiency over competing FPGA- and ASIC-based systems, making it extremely well-suited for real-time, low-power edge applications.

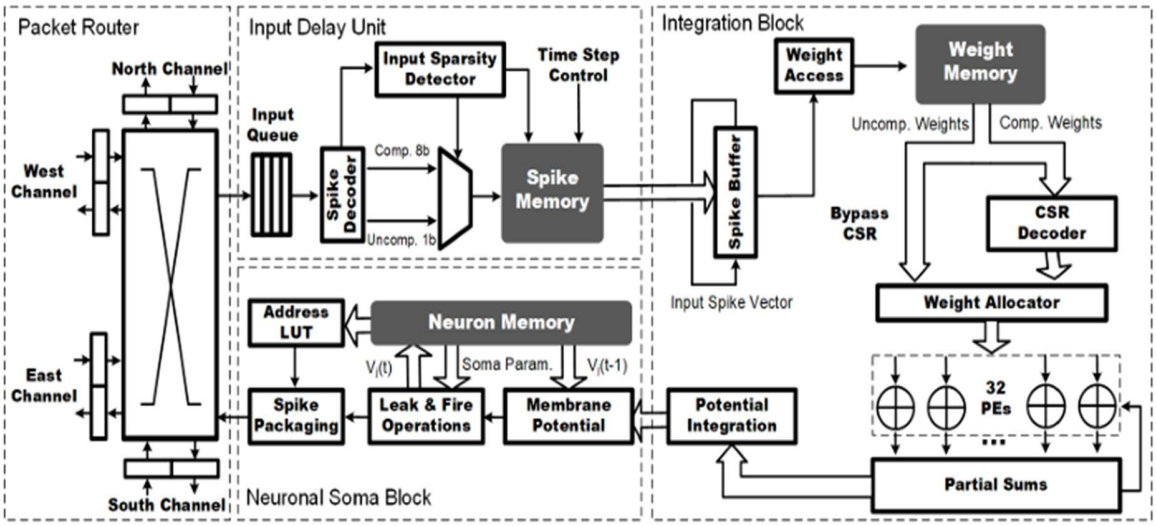

Fig.2.7 Architecture of a single core in ESSA.[6]

2.3 FPGA vs ASIC implementation

The quest for computational efficiency emulating the brain's behavior has driven the interest in Spiking Neural Networks (SNNs), a third-generation neural network model that mimics the sparse and event-driven communication seen in biological neurons. In contrast to traditional artificial neural networks (ANNs) with continuous activation functions and fixed time steps, SNNs take advantage of temporal dynamics and the spiking behavior of neurons, thus allowing processing of information with accurate timing and low energy consumption. Hardware implementation of SNNs is critical to unleash their full potential in real-world applications, including robotics, edge computing, and neuromorphic engineering. Among the most notable hardware platforms to implement SNNs are Field Programmable Gate Arrays (FPGAs) and Application-Specific Integrated Circuits (ASICs). Although each platform has its advantages and limitations, the selection between them has a significant influence on the efficiency, scalability, and tractability of SNNs across domains.

FPGAs are semiconductor chips that are capable of being reconfigured at the hardware level after manufacturing. Their most appealing aspect is this reconfigurability, especially in prototyping and academic research contexts. FPGAs enable quick development and iterative designs, allowing engineers to try out various SNN topologies, learning rules, and communication protocols without paying the cost and time involved in silicon fabrication. The parallel nature of FPGA structures accommodates the simultaneous activities of SNNs, including spike generation, synaptic integration, and updating membrane potentials. In addition, FPGAs allow time-multiplexing and sharing resources, making possible simulations of large-scale SNNs even when logic resources are limited. Open-source software and synthesis platforms at a high level decrease further the SNN implementation complexity on FPGAs. Nevertheless, FPGAs are generally limited by restricted on-chip memory and power inefficiencies due to their general-purpose nature. The programmable interconnects, although very flexible, add delay and power overhead in contrast with ASICs, which restricts the usage of FPGAs for ultra-low-power or real-time applications.

In contrast, ASICs are a dedicated hardware solution designed for a particular application that provides better energy efficiency, performance, and area optimization. ASIC-based implementations of SNNs are best suited for production system deployment where power and latency are essential, for example, mobile devices, autonomous drones, and biomedical implants. By adapting the data path, memory structure, and communication fabric, ASICs can be tailored to fit the sparse, event-based character of SNNs. This leads to vastly lower dynamic and static power draw, which is a fundamental requirement for battery-powered systems. ASIC-based SNN platforms can provide enormous parallelism, sub-milliwatt power usage, and real-time execution. ASIC development, though, involves high non-recurring engineering (NRE) expenses, lengthy design times, and reduced flexibility after fabrication. These characteristics render them less suitable for fast-changing research areas or applications requiring frequent algorithmic renewals or architectural modifications.

The SNNs' computational model further complicates the trade-off between FPGA and ASIC implementations. FPGAs, with their configurable logic blocks and clock management resources, can accurately model these intricate dynamics, providing researchers with a sandbox for investigating novel models and learning mechanisms. They also enable mixed-signal designs and interfacing with external sensors for real-time sensory processing. But the intricateness of SNNs tends to cause resource contention, routing congestion, and timing violations in FPGA designs, particularly at scaling up towards biologically relevant networks. ASICs can resolve these issues by integrating hardware accelerators and neuromorphic cores that inherently support spiking behavior, programmable synapses, and local learning engines. Custom silicon is able to implement mixed-signal neurons and specialized SRAM blocks to simulate synaptic delays and membrane potentials with greater precision and lower power than FPGAs.

Another key issue in assessing FPGA and ASIC implementations of SNNs is scalability. Although FPGAs offer an adaptable platform for small- to mid-scale SNNs, their scalability is naturally limited by dedicated silicon area, availability of logic blocks, and routing infrastructure. Deep SNN applications like real-time vision processing or cognitive robotics

tend to demand networks with billions of neurons and trillions of synapses—scales that are difficult to implement on a single FPGA. But ASICs can be constructed with compact neuromorphic cores connected by asynchronous networks-on-chip (NoC), promoting gigantic SNNs while keeping power consumption in check. Methods like hierarchical memory design, spike compression, and event-driven scheduling are better supported in ASICs because they can optimize each transistor for the workload in question. However, ASIC scalability is limited by silicon cost of manufacture and the challenge of mapping heterogeneous SNN models onto an inflexible hardware platform. This trade-off makes the role of hybrid solutions, where FPGAs are employed in prototyping and ASICs in product deployment, more important.

In conclusion, FPGA or ASIC implementation of SNNs depends on the application, development stage, and resources available. FPGA provide unrivalled flexibility, reconfigurability, and fast prototyping, and as such, are critical to academic and early-stage research. Its applicability to testing new SNN architectures and learning paradigms makes it more flexible. On the other hand, ASICs provide unparalleled performance-per-watt, small form factor, and long-term deployment reliability, making them the go-to solution for commercial-grade neuromorphic systems and energy-limited settings. As the SNN community matures, co-design methods that capitalize on the abilities of both FPGAs and ASICs could become standard. Such hybrid workflows would speed up the algorithm-to-silicon development cycle, making brain-inspired intelligence a practical reality for a range of edge and embedded systems.

3

METHODOLOGY

3.1 Neuron

A neuron is a specialized nervous system cell that is responsible for conducting information around the body. The structure of a neuron consists of three predominant components: dendrites, the soma (or cell body), and the axon. Dendrites are branching projections responsible for receiving signals from other neurons and bringing this information towards the soma. The soma houses the nucleus and is tasked with processing the incoming signals. When the signals that are accumulated surpass a particular threshold, the neuron produces an electrical impulse called an action potential. The impulse propagates along the axon, a long thin extension, to the axon terminals. The terminals make contact with other neurons, muscles, or glands and release neurotransmitters that transfer the signal to the subsequent cell. This process enables neurons to talk quickly and accurately, enabling a high degree of coordination among a great many functions ranging from muscle movement to intellectual functions such as decision-making and memory.

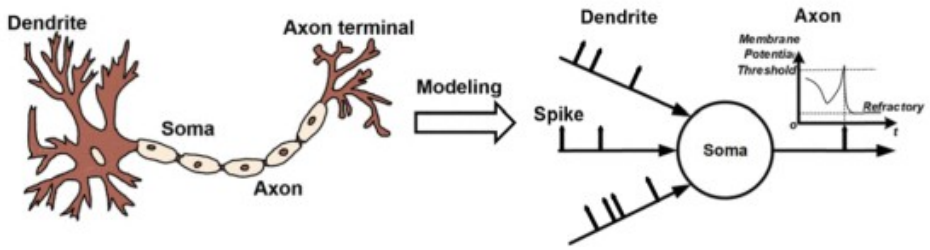

Fig 3.1 Neuron Model [23]

Aside from their biological roles, neurons are the basis for computational models utilized in artificial intelligence and neuroscience. Such models describe the intricate behavior of biological neurons in abstract forms that extract the key processes of signal transfer. In a standard computational model, inputs are summed at the central node that would represent the soma. If the summed inputs exceed a predetermined threshold, the model creates a spike or output signal, simulating the discharge of an actual neuron. After discharge, the model has a refractory period when it is temporarily unable to discharge again—just like the biological reset process. These models assist scientists in simulating neural networks, understanding brain activity, and creating intelligent algorithms with capabilities for learning and decision-making. In either biological or artificial contexts, the efficiency with which the neuron processes and communicates information makes it an essential building block for both the nervous system and contemporary computational systems.

3.1.1 Spiking Neural Networks

Spiking Neural Networks aim is to bridge traditional artificial intelligence (AI) approaches with the rules of biological neural processing observed in the brain. Unlike the traditional neural networks, which work on continuous values to process information, SNNs work on discrete events, or spikes, for information transmission and processing, thereby possessing a more biologically realistic and energy-efficient computational model.

Artificial neural networks, like feedforward networks, convolutional neural networks, and recurrent neural networks, operate by employing continuous activation functions like sigmoid, tanh, or ReLU. Biological neurons, on the other hand, encode information in the form of short, sudden voltage spikes—referred to as spikes—that are transmitted through synapses.Spiking Neural Networks are created to mimic this functionality. Data in SNNs is represented in the time and rate of spikes rather than just the signal size. Temporal representation allows temporal dynamics to be included, so the network can better process spatiotemporal patterns.

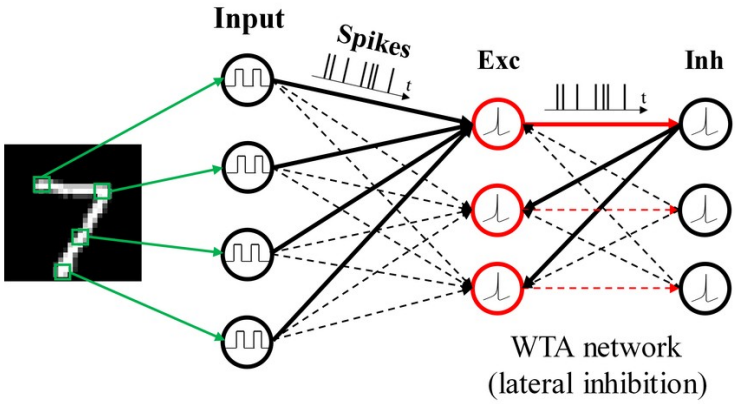

Fig 3.2 SNN Methodology [24]

3.2 Neuron Models

Spiking Neural Networks (SNNs) aim to mimic the information processing mechanisms of the brain more closely than traditional artificial neural networks. The central component of SNNs is the **spiking neuron**, which transmits and processes information using discrete electrical pulses called **spikes**. The mathematical models that describe how these neurons behave are known as **spiking neuron models**.

These models vary in complexity and realism, ranging from simple integrator-based equations to detailed simulations of ion channel dynamics. This article explores the most commonly used spiking neuron models:

1. Integrate-and-Fire (IF) Model
2. Leaky Integrate-and-Fire (LIF) Model
3. Izhikevich Model
4. Hodgkin-Huxley Model

3.2.1 Integrate and Fire Model

The **Integrate-and-Fire** model is one of the earliest and simplest neuron models. It abstracts a neuron as a capacitor that integrates incoming currents over time. When the membrane potential reaches a threshold, the neuron fires a spike and resets.

$$\frac{dV(t)}{dt} = \frac{I(t)}{C}$$

- $V(t)$: Membrane potential
- $I(t)$: Input current
- C : Membrane capacitance

Spiking condition: When $V(t) \geq Vth$, the neuron emits a spike and $V(t)$ is reset to $Vreset$.

Features

- Easy to simulate
- No biological accuracy for sub-threshold dynamics
- Lacks membrane leakage (idealized)

3.2.2 Leaky Integrate and Fire Model

The **Leaky Integrate-and-Fire (LIF)** model improves on the IF model by incorporating a "leak" to the membrane potential, simulating the passive ion channel behavior of biological membranes. It remains one of the most widely used models in computational neuroscience.

$$\tau_m \frac{dV(t)}{dt} = -[V(t) - V_{rest}] + R.I(t)$$

Where

$\tau_m = R\,.\,C$ (Membrane time constant)

R = Membrane resistance

V_{rest} = Resting membrane potential

$I(t)$ = Input current

When $V(t) \geq V_{th}$, the neuron spikes and $V(t)$ is reset.

Features

- More biologically realistic than IF
- Incorporates passive decay
- Still lacks refractory period and spike shape

Advantages

- Simple, computationally efficient
- Captures basic spiking behavior
- Well-suited for neuromorphic hardware

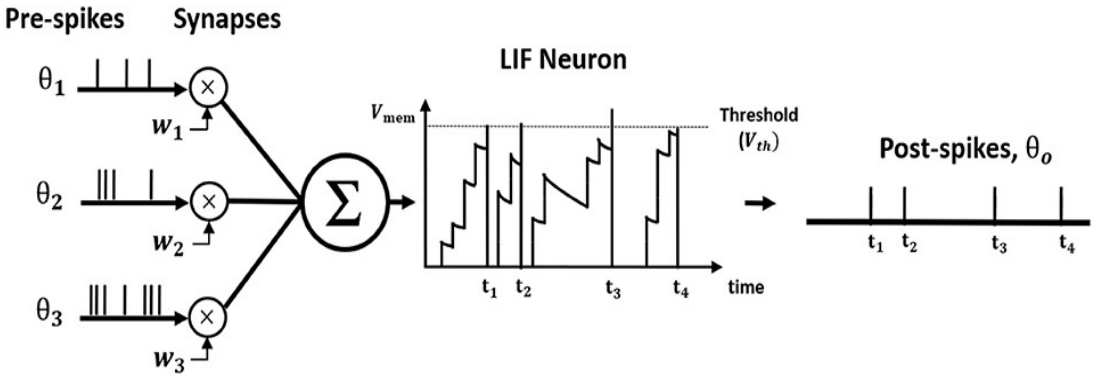

Fig 3.3 LIF Neuron Working principle [25]

3.2.3 Izhikevich Model

Developed by Eugene Izhikevich, this model strikes a balance between biological realism and computational efficiency. It captures a wide range of neuronal dynamics, such as bursting, tonic firing, and adaptation.

$$\frac{dv}{dt} = 0.04v^2 + 5v + 140 - u + I$$

$$\frac{du}{dt} = a(bv - u)$$

With reset conditions

$$\text{if } v \geq 30 \text{ mV, then } \begin{cases} v = c \\ u = u + d \end{cases}$$

where,

v = Membrane potential

u = Membrane recovery variable

a, b, c, d = parameters controlling dynamics

Neuron Type	**a**	**b**	**c**	**d**
Regular Spiking	0.02	0.2	-65	8
Fast Spiking	0.1	0.2	-65	2
Intrinsically Bursting	0.02	0.2	-55	4

Table 3.1 Parameters for controlling spiking neuron

Features

- Reproduces 20+ known spiking patterns
- Efficient (100x faster than Hodgkin-Huxley)
- Parameter tuning required

Use Cases

- Large-scale brain simulations
- Neuron-type modeling
- Robotics and neuromorphic systems

3.2.4 Hodgkin-Huxley Model

The **Hodgkin-Huxley model** is the gold standard in biophysical realism. Developed in 1952 based on studies of squid giant axons, it models ionic mechanisms (Na+, K++, and leakage channels) that generate action potentials.

$$C_m \frac{dV}{dt} = I_{ext} - (I_{Na} + I_K + I_L)$$

Where

V = Membrane Potential

$I_{Na} + I_K + I_L$ = Total ion currents

Features

- Most biologically accurate spiking model
- Describes full spike shape and refractory behaviour
- Requires solving nonlinear differential equations

Limitations

- Computationally expensive
- Parameter-intensive
- Complex to tune

Use Cases

- Detailed neuron modelling
- Pharmacological studies
- Validating simplified models

Feature	IF	LIF	Hodgkin-Huxley	Izhikevich
Biological Realism	Low	Moderate	Very High	High
Spike Shape Modelled	No	No	Yes	Approximate
Refractory Period	Manual	Manual	Yes	Yes
Complexity	Very Low	Low	High	Moderate
Computational Cost	Minimal	Low	High	Moderate
Burst & Adaptation	No	No	Yes	Yes
Customization	Low	Moderate	Very High	Very High

Table 3.2 Comparison between Different types of spiking Neuron Models

3.3 SPIKE TIME DEPENDENT PLASTICITY

Spike-Timing-Dependent Plasticity (STDP) is a biological learning rule applied in neuroscience and computational brain models to describe how synaptic efficacy (between neurons) is modified depending on the timing of spikes (electric impulses). STDP is a type of Hebbian learning, encapsulated by the slogan: "cells that fire together, wire together"—with an important caveat: the precise timing of firing makes a difference.

In STDP, synaptic weight changes as function of relative spike timing between spikes of the presynaptic neuron (signal-sending) and the postsynaptic neuron (signal-receiving). There are two primary cases:

1. Potentiation (Synaptic Strengthening):

If presynaptic neuron is activated just before postsynaptic neuron (typically in a timeframe of 20 milliseconds), the synaptic weight is incremented. This indicates that the presynaptic activity was responsible for the postsynaptic firing, thus strengthening the connection.

2. Depression (Weakening the Synapse):

When the presynaptic neuron fires subsequent to postsynaptic neuron, connection is attenuated. This suggests the presynaptic spike was less meaningful or otherwise delayed to affect the postsynaptic response.

The synaptic change usually takes an exponential form—most change happens when spike times overlap and tapers off as the interval becomes larger.

STDP has also been found in other areas of the brain, including the hippocampus and neocortex. It is considered important for learning, memory consolidation, and development of neural networks. It enables neural systems to learn cause-effect relationships and temporal patterns, which is how humans link events that happen near each other in time.

In artificial neural networks, STDP has motivated more biologically plausible learning rules applied in spiking neural networks (SNNs). Such models try to better simulate real neuronal dynamics than conventional artificial neural networks, employing real spikes (discrete events) instead of continuous activations.

Mathematically it is given as follows:

$$\Delta w = \begin{cases} -A_{-}e^{-\Delta t/\tau_{+}}, & \Delta t < 0 \\ A_{+}e^{\Delta t/\tau_{-}}, & \Delta t \geq 0 \end{cases}$$

Where:

$\Delta t = t_{post} - t_{pre}$
A_{-}, A_{+} are the learning rates
τ_{+}, τ_{-} are time constants

The phenomenon of STDP is described as a biphasic exponentially decaying function. That is, the instantaneous change in weights is given by

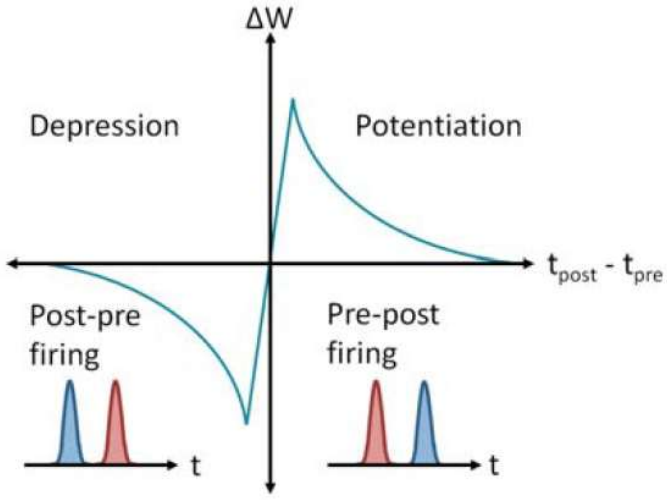

Fig 3.4 STDP Graph for weights update [26]

Based on the spike pulse, the weight memory gets changed with respect to the learning rates and time constants. These values are generated by LUT approximations of exponential functions, which were helpful in computations to generate weights for each image it used for learning. This helps in reducing complexities of convolutional computing which takes lot of memory and time to learn.

3.4 IMAGE ENCODING TECHNIQUES

3.4.1 Rate Encoding

Rate coding is the most common and most basic encoding scheme in SNNs. Here, the value or intensity of a pixel is encoded by the rate or frequency with which its respective neuron emits spikes. High-value or bright pixels induce neurons to emit spikes with higher frequency within a given time interval, whereas darker pixels emit spikes at lower frequency. For example, a white pixel could produce 20 spikes in 100 ms, but a black one could produce none. This approach is simple and robust, so it is appropriate for most SNN applications. But it is not very energy efficient, since a lot of spikes are needed to represent information.

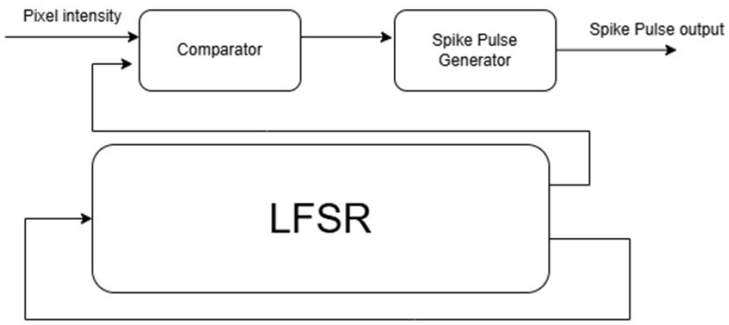

Fig 3.5 Rate encoding technique circuit

3.4.2 Temporal Encoding

Temporal coding focuses on precise timing of spikes rather than rate. In this scheme, each pixel fires only once, and the time of firing is inversely proportional to its intensity—the brighter the pixel, the earlier the spike. For example, a pixel with maximum brightness may spike at 1 ms, while one with half brightness may spike at 10 ms. This encoding method produces very sparse spike trains, which makes it energy-efficient and suitable for real-time, low-power systems.

However, it is more sensitive to noise and requires precise control of spike timing, which can be difficult to implement in practice. It is also called as Time to First Spoke (TTFS) encoding method.

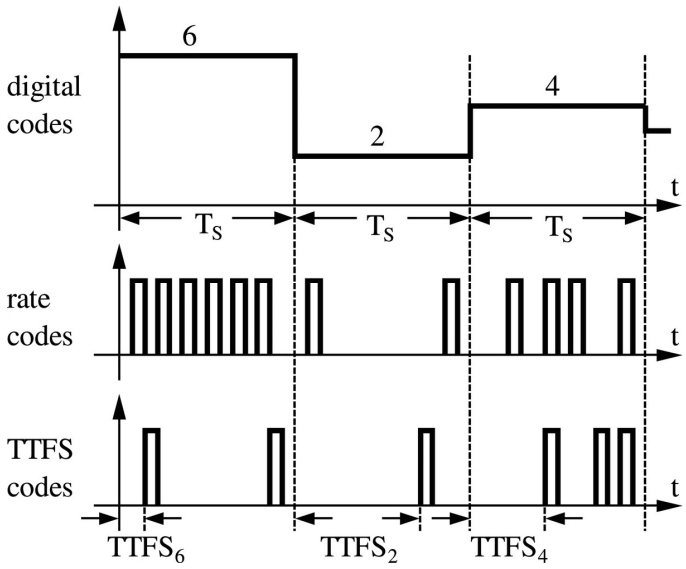

Fig 3.6 Waveform showing Image pixel encoding. [27]

3.5 ARCHITECTURE

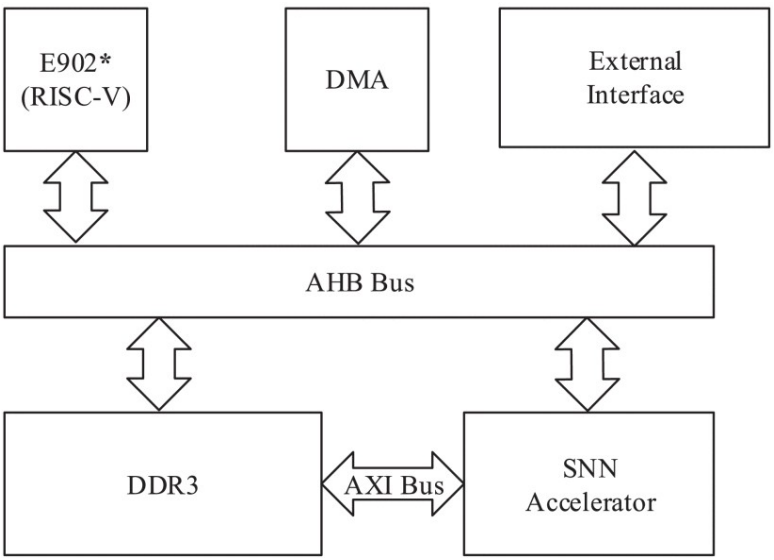

Fig 3.7 Architecture on FPGA

The diagram represents a system-on-chip (SoC) architecture optimized for accelerating Spiking Neural Network (SNN) computations, integrating various hardware units coupled with standardized bus protocols. The heart of the system is the Advance High-performance Bus (AHB), an industry-standard interconnect enabling communication between main processing and data movement units. The RISC-V core, a small footprint, low-power processor, handles general control, instruction sequencing, and high-level SNN management such as preprocessing and interfacing functions. It interacts with other subsystems through the AHB bus. The Direct Memory Access controller runs independently of CPU and facilitates fast, high-speed data movement between memory and peripherals (for example, from DDR3 memory to the SNN accelerator) without using processor resources. This isolation of data movement and computation maximizes throughput and reduces latency. The External Interface enables the system to interface with external devices or networks and facilitates input/output activities like sensory data ingestion or result output transmission.

One of the major pieces of this architecture is the SNN Accelerator, a specialized hardware block specifically designed to perform the computationally demanding operations in SNN inference and learning—e.g., spike generation, synaptic weight update, and membrane potential integration. The accelerator is connected to the DDR3 memory via the Advanced eXtensible Interface (AXI) Bus, which offers high-bandwidth and low-latency memory access essential for massive-scale SNN operations. DDR3 memory serves as the main data storage hub, housing neural parameters, spike event queues, and intermediate results. The use of separate AHB and AXI buses allows for an effective division of workload: control signaling and lightweight communication are handled over AHB, while high-speed data movement necessary for SNN processing occurs over AXI. This hierarchical and modular architecture allows for scalability and modular growth, which supports a power-efficient and flexible SNN hardware platform. In general, this architecture is appropriate for real-time embedded neuromorphic applications that need low power and close integration between control logic and neural computing.

3.5.1 SNN Accelerator

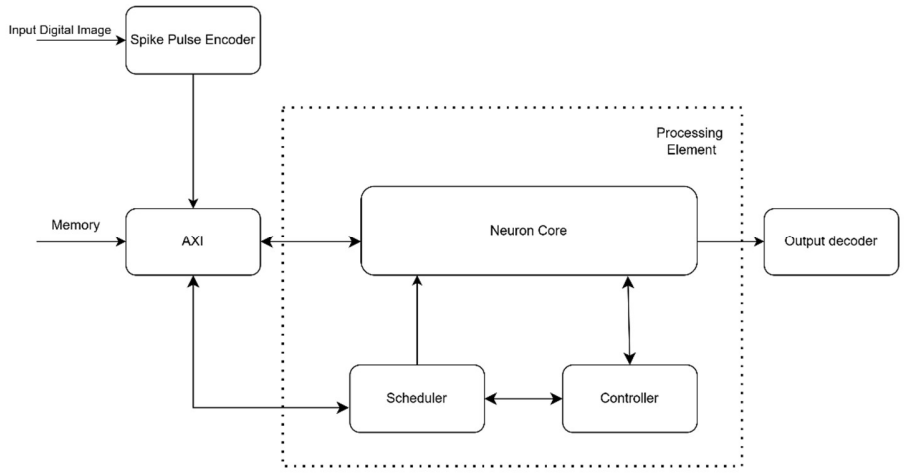

Fig 3.8 SNN Accelerator Architecture

The Architecture for a hardware realization of a Spiking Neural Network (SNN) that is intended to process input digital images based on principles of neuromorphic computing. At the input stage, digital images are initially passed through a Spike Pulse Encoder, which maps the pixel intensity values of the digital image into corresponding spike trains—modelling how biological sensory systems represent visual stimuli. This encoded spike information is then routed through AXI bus, which is used to enable high-speed communication of data among hardware components. The information about spikes then gets to the Synaptic Core, whose synaptic weights scale the spike signals before transmitting them to the Neuron Core. The Synaptic Core mimics biological synapses by scaling signal strength according to stored weights and learning rules. Spike accumulation and membrane potential dynamics are managed within the Neuron Core. Upon a neuron reaching a specific threshold, it outputs a spike to the subsequent layer, simulating biological neuron behaviour.

Supporting components have key roles in managing and optimizing the computation. The Scheduler schedules spike events for execution and enforces timing throughout different stages of processing to keep all components synchronized. The Controller regulates system-level processes, for example, to initialize the network, control learning rules (Spike-Timing-Dependent Plasticity),

and mediate communication with other systems or processors. After processing the spikes in the Neuron Core, the outputs are directed to the Output module, where spike patterns are translated or transformed into an understandable response—like a classification tag or action signal. This module-based hardware design ensures parallelism, low power, and real-time processing, which it is best suited for edge AI, robotics, and brain-inspired computation. The architecture showcases a highly synchronized system where spike-based encoding, synaptic weighing, and neuron firing are unified in a lean pipeline for efficient and scalable SNN inference.

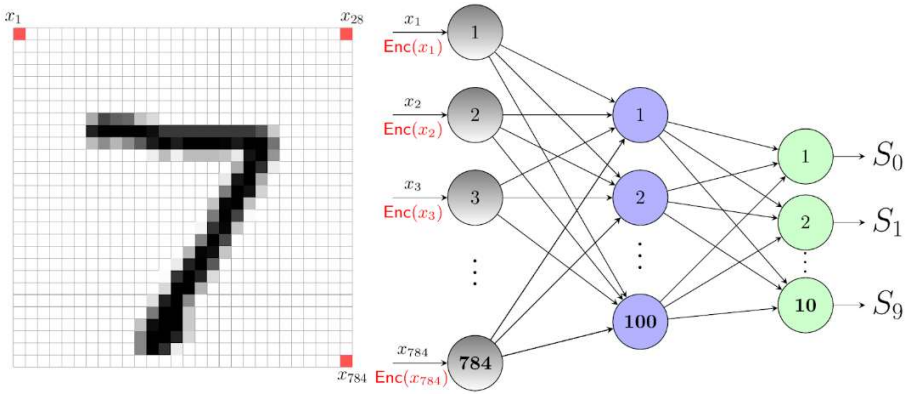

Fig 3.9 Internal Structure of Neuron Core [28]

Since the images used for the classification are of size 28x28 pixels the SNN network contains four layers. In the first layer it contains 784 neurons in next hidden layer contains 100 neurons and in the output layer 10 neurons for the classification. In total it contains 894 neurons. By giving the output to the decoder circuit, we can observe the image classified.

4

HARDWARE IMPLEMENTATION

4.1 HARDWARE TOOLS

4.1.1 FPGA

In the realm of digital circuits Field-Programmable Gate Arrays, are a particular kind of integrated circuit (IC) that provide special benefits. In contrast to conventional CPUs or GPUs, which are pre-programmed with a predefined design. FPGAs are programmable to carry out particular digital tasks after they are manufactured. Their flexibility to be programmed makes them adaptable instruments for a range of uses.

Structure:

They are made up of a variety of programmable interconnects that encircle a grid of customizable logic blocks (CLBs). Consider CLBs as tiny components that are capable of simple logic operations, and consider interconnects as wires that join them to form more intricate circuits.

Programmability:

By submitting a setup bitstream, the user establishes the FPGA's functionalities. The operation and interconnection of the CLBs are fundamentally set by this bitstream.

Advantages:

Hardware Customization:

FPGAs let you create unique circuits for hardware that are suited to your own requirements. When high performance is needed or current off-the-shelf solutions aren't appropriate, this can be helpful.

Flexibility:

FPGAs are perfect for situations where the hardware requirements may change over time since they can be reconfigured in the field. Without changing the chip itself, you can upgrade the functionality.

Performance:

By implementing the algorithms directly in hardware, FPGAs that can achieve exceptional efficiency for certain jobs, possibly outperforming solutions-based software on general-purpose processors

4.1.2 Types of Boards

Different types of boards available in market. Different FPGA boards are used for different applications. Due to different sizes and number of components available on the board, the power budget varies from board to board. Basys3 FPGA board, which is built around Xilinx Artix-7 XC7A35T chip, has low power consumption and can be used to execute soft-core processors such as MicroBlaze. Power usage on the board can be divided into static (leakage) and dynamic (switching activity) power. Static power is typically 20–40 mW, whereas dynamic power is a function of clock frequency, logic usage, and I/O activity. A basic MicroBlaze configuration with UART and GPIO operating at 50–100 MHz uses around 100–200 mW. With additional peripherals such as memory controllers, SPI, or timers, overall usage can reach 300–500 mW or more. In a scenario of full utilization with active I/O and peripherals, the power budget can approach 1 W. Power may be supplied through USB or an external 5V source, and it is possible to measure in the real world using a multimeter on the input rails.

To accurately estimate or analyse power consumption, Xilinx provides tools such as the Xilinx Power Estimator (XPE) and Vivado Power Analyzer. XPE offers early design estimates, whereas Vivado offers post-implementation power analysis based on implemented logic and routing. In comparison, other FPGA boards such as the Zed Board (Zynq-7000) or DE10-Nano (Intel Cyclone V) generally consume more power (2–5 W), because of more intricate architectures and built-in processors. In contrast, the Basys3 is efficient with an average total power budget of 100–500 mW for typical MicroBlaze-based designs and is thus best suited for learning purposes and low-power embedded applications.

4.1.3 Basys 3 FPGA Board

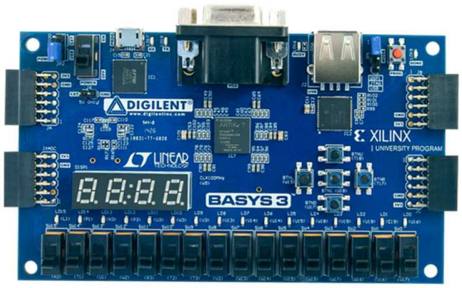

Fig 4.1 Basys3 FPGA Board [29]

The Xilinx Artix-7 family is constructed on a 28nm low-power CMOS process, providing a balance of performance, power consumption, and cost. It is optimized for a broad set of applications ranging from embedded systems, digital signal processing, and communications. Artix-7 includes a scalable architecture with logic cells made up of 6-input Look-Up Tables (LUTs), carry chains and flip-flops. These logic blocks are grouped into Configurable Logic Blocks (CLBs), which enable fast implementation of complicated digital functions. Artix-7 devices also feature Block RAM (BRAM), DSP slices, and clock management tiles for sophisticated timing control.

High-speed I/O is supported by the architecture through support for LVDS, DDR3 memory, and PCIe interfaces, which positions it well for use in data-intensive applications. The Artix-7 also features XADC (analog-to-digital converter), multiple clock domains, and solid power management features. With soft processor support integrated into the chip, the Artix-7 provides flexible, reconfigurable computing solutions on a low-power platform.

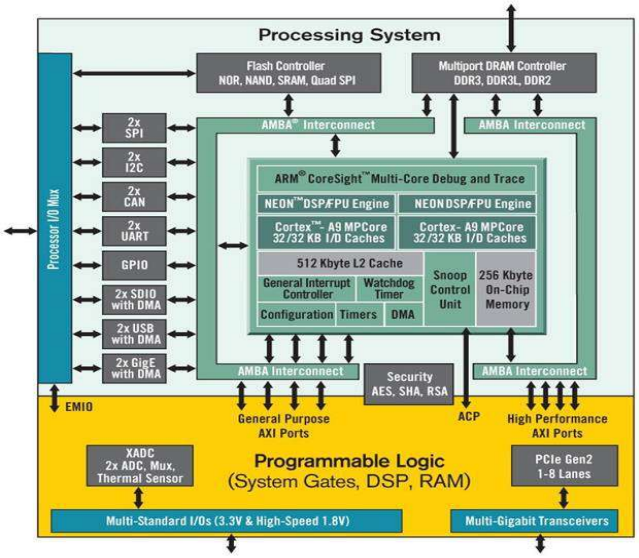

Fig 4.2 Architecture of Xilinx Artix 7 FPGA [30]

4.1.4 AXI Protocol

A crucial component of the ARM AMBA standard, the AXI protocol is made especially for high-performance, high-frequency system-on-chip (SoC) designs. It makes communication easier between peripherals, memory controllers, and CPUs among other SoC components.

Features:

High Speed and Minimal Latency:

Because AXI can carry data at a fast speed, it can be used in high-performance applications.

Five Separate Channels:

Read Address Channel: It carries the address for read operations.

Read Data Channel: The data being read in this channel.

Write Address Channel: It carries the address for write operations.

Write Data Channel: It carries the data to be written.

Write Response Channel: It carries the response to write operations.

Phases for data and addresses separately:

More flexibility and efficiency in data transfer are possible when the address and data phases are separated.

Assistance with Several Masters and Slaves:

Multiple masters and slaves can be managed by AXI, allowing for sophisticated SoC architectures with a variety of interconnected components.

Fig 4.3 AXI protocol

4.2 VIVADO SOFTWARE

Vivado is developed by XILINX, it is used to design and implement complex digital systems on FPGA. Now a days more advancements in vivado so that it can perform Machine learning applications

Features:

The IDE, or integrated design environment, integrated development environment (IDE) for design entry, simulation, synthesis, implementation, and debugging is offered by Vivado.

For automation and scripting, the IDE offers both a (CLI) and a (GUI).

Intellectual Property Integrator: Using pre-built IP cores and facilitating the building of intricate designs with drag-and-drop capabilities are two benefits of the IP Integrator tool

Building custom systems is made simple by its support for third-party IP cores as well as /Xilinx cores.

The Vivado Simulator

A comprehensive simulation platform that facilitates both temporal and functional simulations. The Vivado Simulator facilitates the validation of designs at various phases of development, guaranteeing accuracy prior to execution.

Debugging and Design Analysis:

Advanced analytic capabilities including power analysis, time analysis, and resource use reports are offered by Vivado.

In-system debugging and verification are made easier by integrated debugging tools like the (VIO) and (ILA).

Design flow:

Design entry: Design is entered to the software with the help of HDL like Verilog

Simulation: By using testbenches, output of the design is simulated in waveforms.

Synthesis: it converts the top level design into gate level design is optimizes the design for the targeted FPGA

Implementation: It includes placement routing and bitstream generation

Placemenet: Synthesized logic to specific location on FPGA

Routing: It connects placed logic elements according to the design netlist

Bitstream: All the information of the project is converted into bit stream file in order to dump into hardware

Debugging and Design Analysis:

The functionality of the design is guaranteed using functional simulation.

Timing simulation verifies that, when mapped to the FPGA, the design satisfies timing constraints.Debugging and verification can be done in the real hardware environment with the help of in-system debugging tools

Programming and deployment: Generated bit stream is programmed in hardware manager and it is deployed for testing in the hardware

\

5

RESULTS

5.1 Software Results

```
Predicted_class = 9.0
Actual class = 9
Time for inference = 0.9317989349365234
Accuracy = 98.09160305343512

[Done] exited with code=0 in 267.566 seconds
```

Fig 5.1 Result of Software simulation

5.2 Image encoding Technique
Rate encoder circuit
Schematic

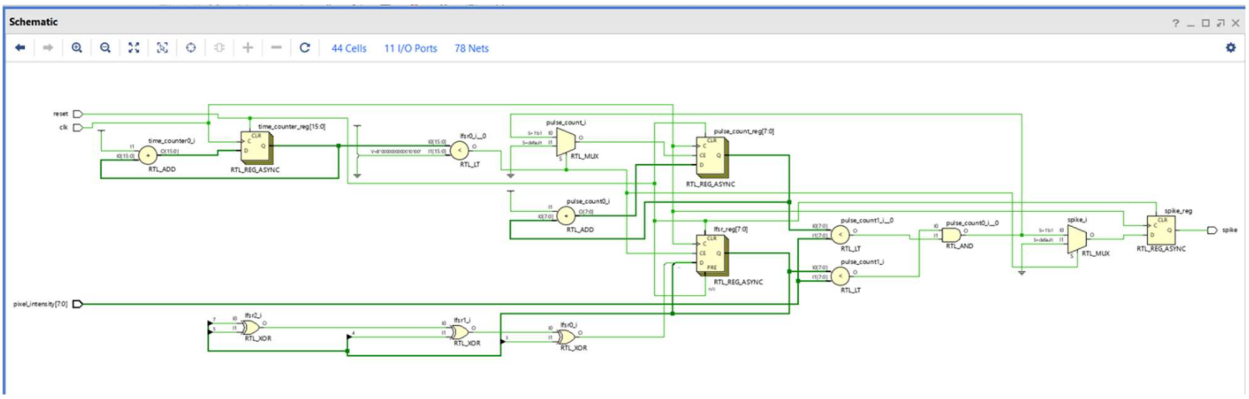

Fig 5.2 Rate encoder circuit Schematic view

Simulation

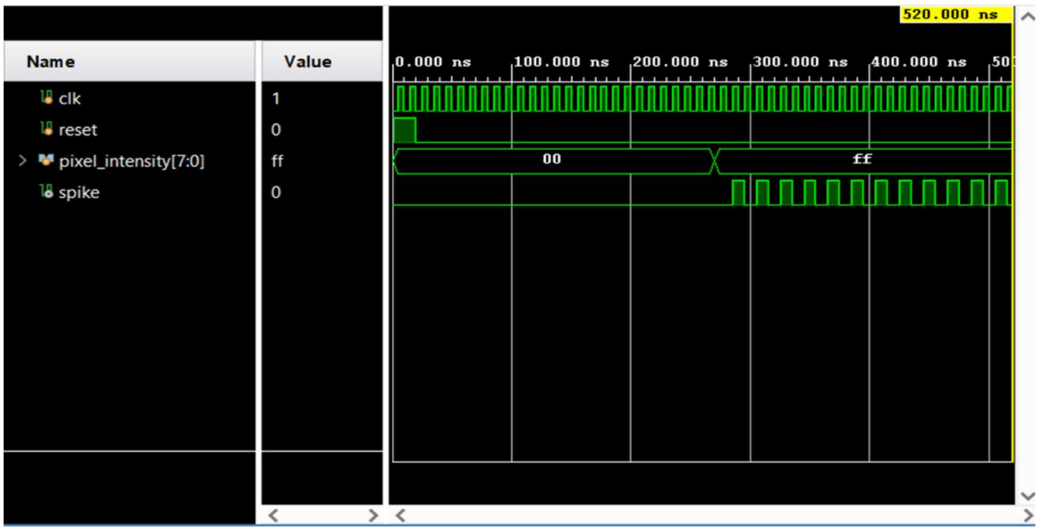

Fig 5.3 Simulation of Rate encoding technique

5.3 Neuron Models

5.3.1Integrate and Fire Model

Schematic

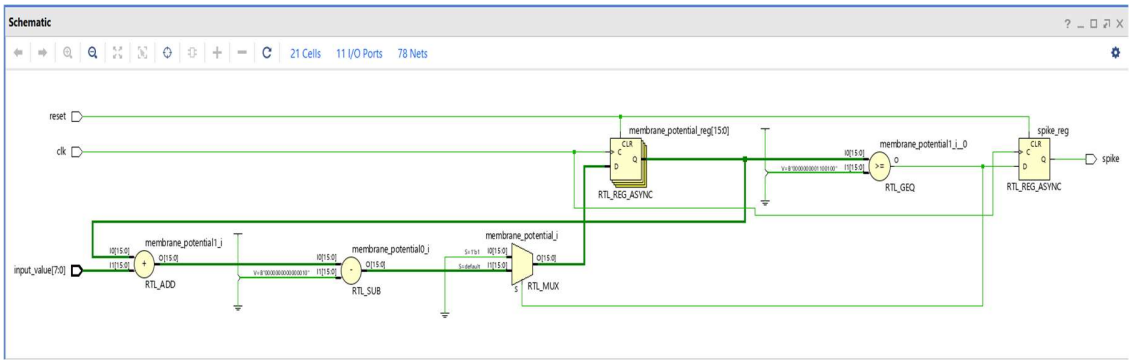

Fig 5.4 Integrate and Fire Neuron Model Schematic view

5.3.2 Leaky Integrate and Fire Model (LIF) Model Design

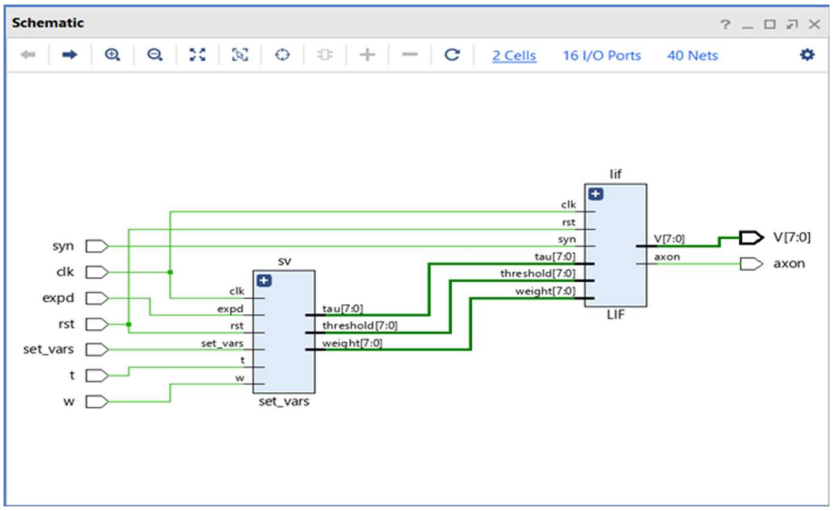

Fig 5.5 LIF Neuron Model

Schematic

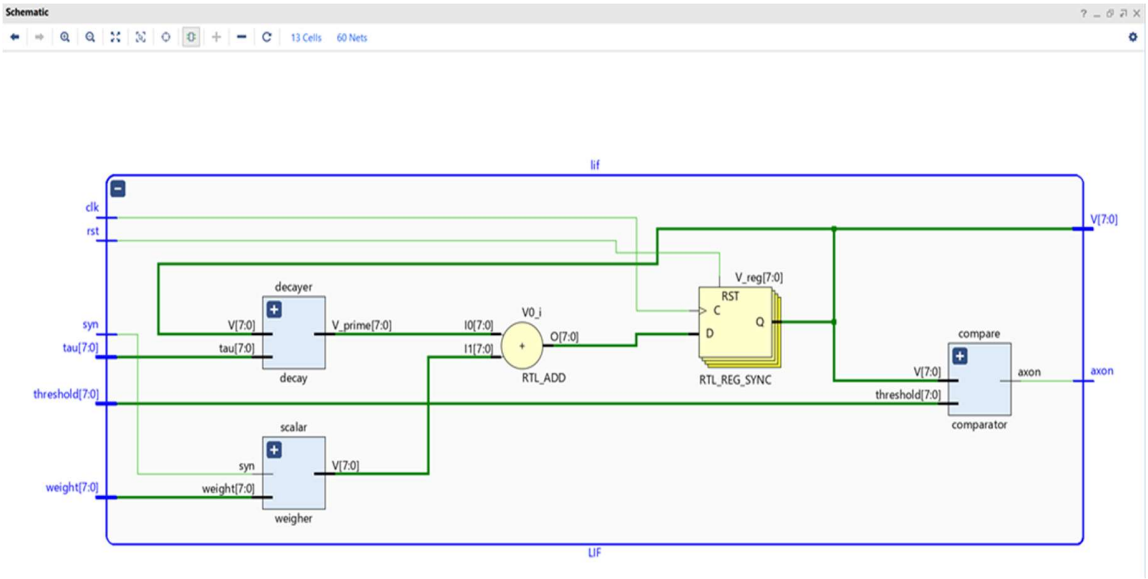

Fig 5.6 Schematic view of LIF Neuron

Simulation result

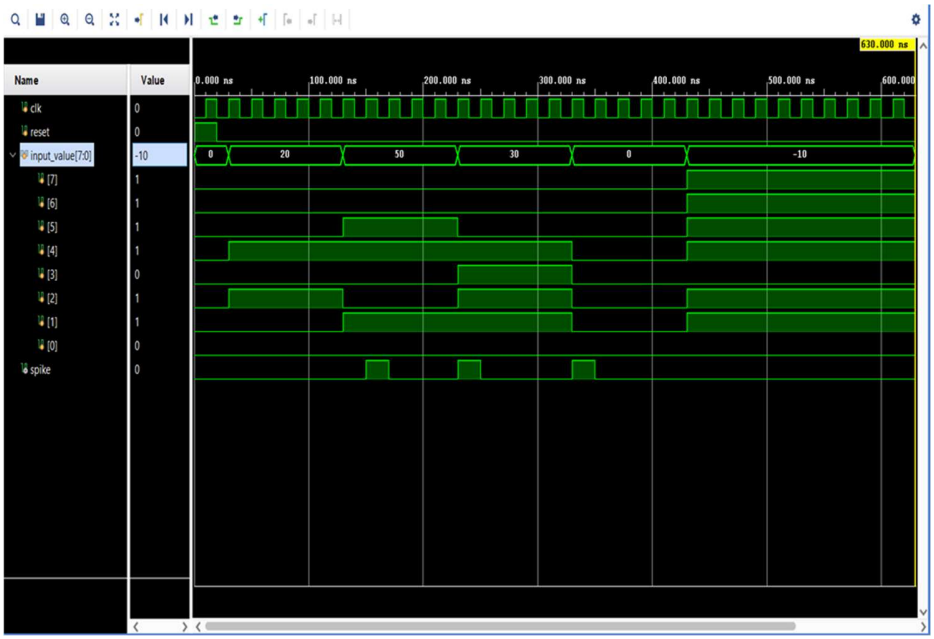

Fig 5.7 Simulation result of LIF Neuron

```
# run 1000ns
Time: 0 | Input Value: 0 | Membrane Potential: 0 | Spike: 0
Time: 30000 | Input Value: 20 | Membrane Potential: 18 | Spike: 0
Time: 50000 | Input Value: 20 | Membrane Potential: 36 | Spike: 0
Time: 70000 | Input Value: 20 | Membrane Potential: 54 | Spike: 0
Time: 90000 | Input Value: 20 | Membrane Potential: 72 | Spike: 0
Time: 110000 | Input Value: 20 | Membrane Potential: 90 | Spike: 0
Time: 130000 | Input Value: 50 | Membrane Potential: 138 | Spike: 0
Time: 150000 | Input Value: 50 | Membrane Potential: 0 | Spike: 1
Time: 170000 | Input Value: 50 | Membrane Potential: 48 | Spike: 0
Time: 190000 | Input Value: 50 | Membrane Potential: 96 | Spike: 0
Time: 210000 | Input Value: 50 | Membrane Potential: 144 | Spike: 0
Time: 230000 | Input Value: 30 | Membrane Potential: 0 | Spike: 1
Time: 250000 | Input Value: 30 | Membrane Potential: 28 | Spike: 0
Time: 270000 | Input Value: 30 | Membrane Potential: 56 | Spike: 0
Time: 290000 | Input Value: 30 | Membrane Potential: 84 | Spike: 0
Time: 310000 | Input Value: 30 | Membrane Potential: 112 | Spike: 0
Time: 330000 | Input Value: 0 | Membrane Potential: 0 | Spike: 1
```

Fig 5.8 Log file of above simulation result

Synthesis result

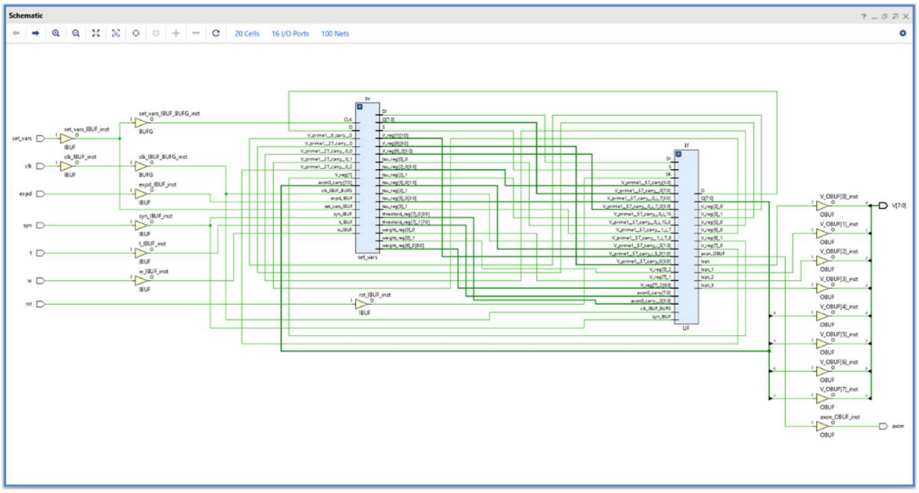

Fig 5.9 Synthesis result of LIF Neuron

5.4 SNN Architeture

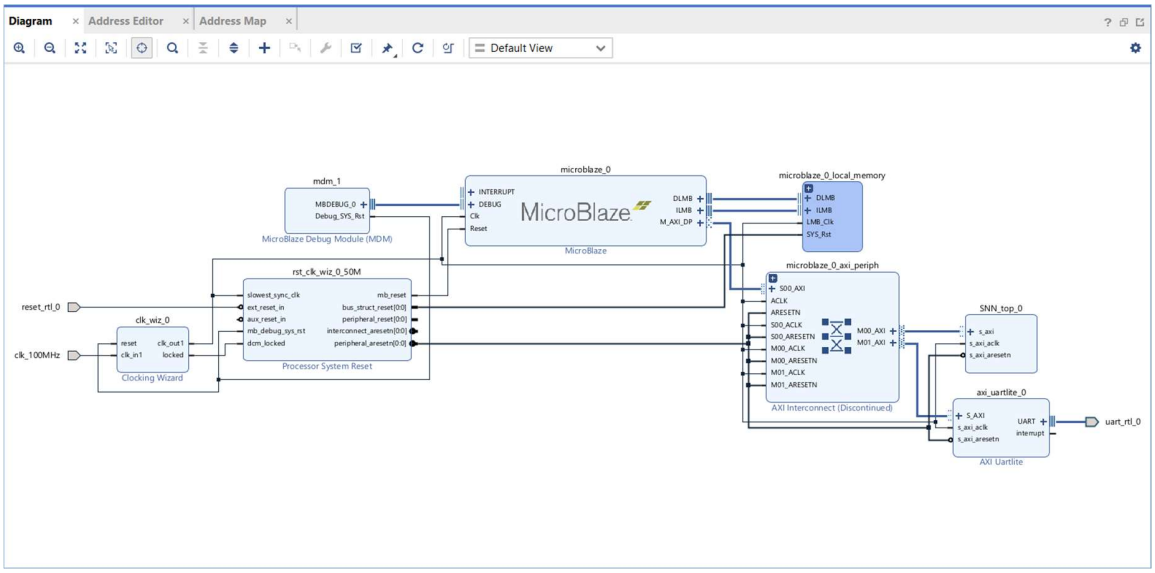

Fig 5.10 SNN Architecture implemented on FPGA

5.5 Reports
5.5.1 Utilization Report

Name	Slice LUTs (63400)	Slice Registers (126800)	F7 Muxes (31700)	F8 Muxes (15850)	Slice (15850)	LUT as Logic (63400)	LUT as Memory (19000)	Block RAM Tile (135)	DSPs (240)	Bonded IOB (210)	BUFGCTRL (32)	MMCME2_ADV (6)	BSCANE2 (4)
design_1_wrapper	10045	3900	1883	713	3240	9820	225	61	3	4	3	1	1
design_1_i (design_1)	10045	3900	1883	713	3240	9820	225	61	3	0	3	1	1
axi_uartlite_0 (desig	92	105	0	0	35	82	10	0	0	0	0	0	0
clk_wiz_0 (design_1_	0	0	0	0	0	0	0	0	0	0	2	1	0
mdm_1 (design_1_m	91	110	0	0	38	84	7	0	0	0	1	0	1
microblaze_0 (desig	1101	781	33	0	432	896	205	0	3	0	0	0	0
microblaze_0_axi_pe	83	109	0	0	46	83	0	0	0	0	0	0	0
microblaze_0_local_	12	14	0	0	9	10	2	32	0	0	0	0	0
rst_clk_wiz_0_50M (d	18	38	0	0	13	17	1	0	0	0	0	0	0
SNN_top_0 (design_	8648	2743	1850	713	2688	8648	0	29	0	0	0	0	0

Table 5.1 Utilized components list

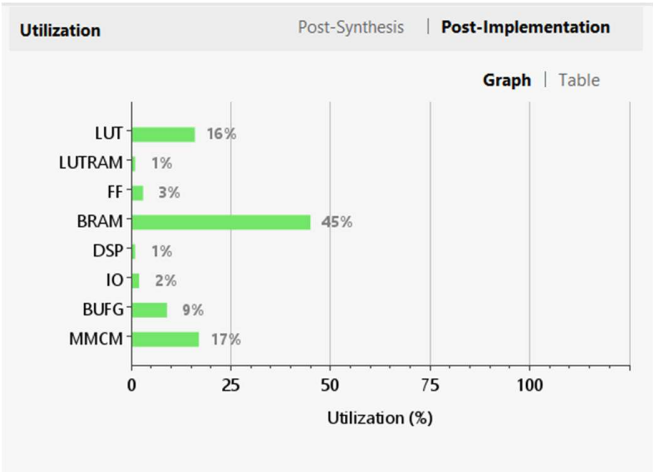

Fig 5.11 Graph showing % of components utilized

5.5.2 Power Report

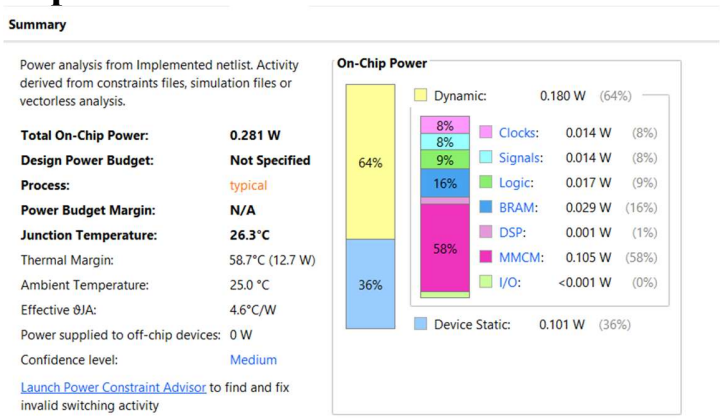

Fig 5.12 Power utilization Report

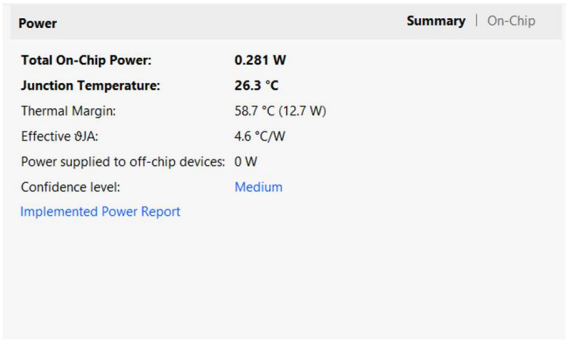
Power
Summary | On-Chip
Total On-Chip Power: 0.281 W
Junction Temperature: 26.3 °C
Thermal Margin: 58.7 °C (12.7 W)
Effective θJA: 4.6 °C/W
Power supplied to off-chip devices: 0 W
Confidence level: Medium
Implemented Power Report

Fig 5.13 Summary of Power consumption

5.5.3 Timing Report

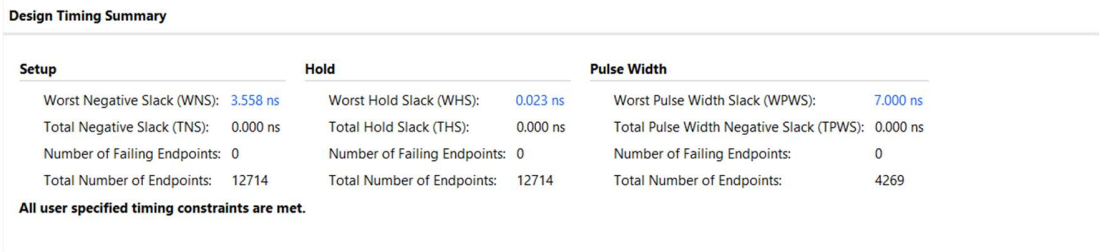
Design Timing Summary
Setup
Worst Negative Slack (WNS): 3.558 ns
Total Negative Slack (TNS): 0.000 ns
Number of Failing Endpoints: 0
Total Number of Endpoints: 12714
Hold
Worst Hold Slack (WHS): 0.023 ns
Total Hold Slack (THS): 0.000 ns
Number of Failing Endpoints: 0
Total Number of Endpoints: 12714
Pulse Width
Worst Pulse Width Slack (WPWS): 7.000 ns
Total Pulse Width Negative Slack (TPWS): 0.000 ns
Number of Failing Endpoints: 0
Total Number of Endpoints: 4269
All user specified timing constraints are met.

Fig 5.14 Timing Report

5.6 Image classification

Input test image1 :

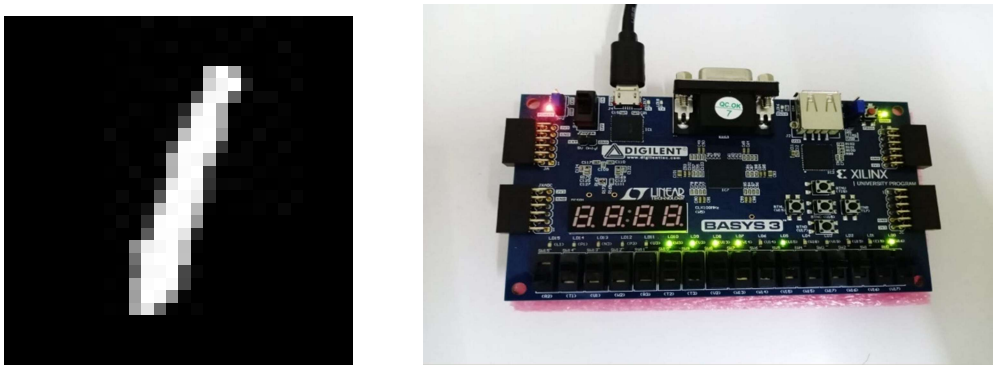

Fig 5.15 Output of the input test image1 on Basys3

Input test image 2:

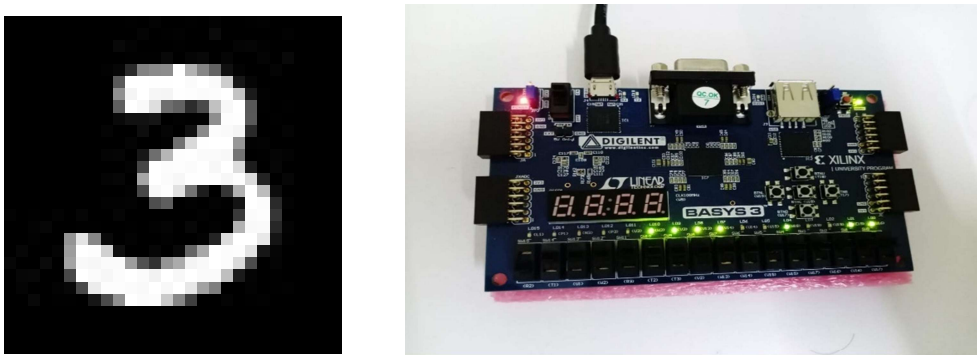

Fig 5.16 Output of the input test image2 on Basys3

Input test image 3:

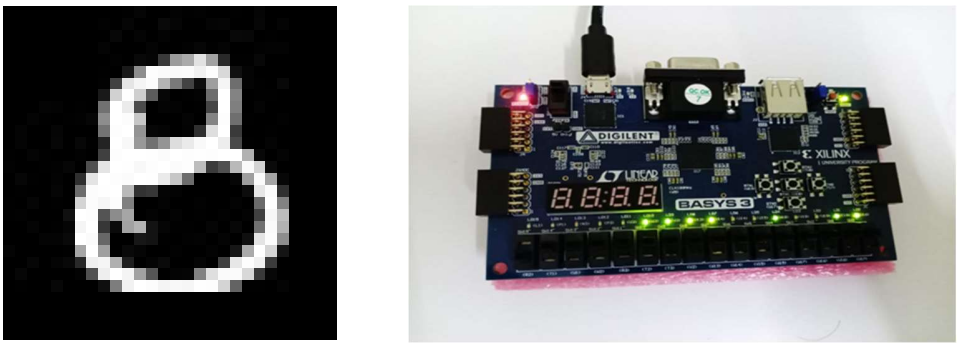

Fig 5.17 Output of the input test image3 on Basys3

Input test image 4:

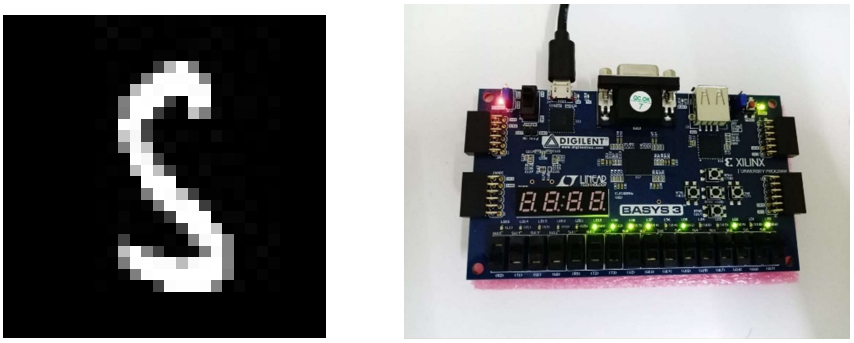

Fig 5.18 Output of the input test image4 on Basys3

Input test image 5:

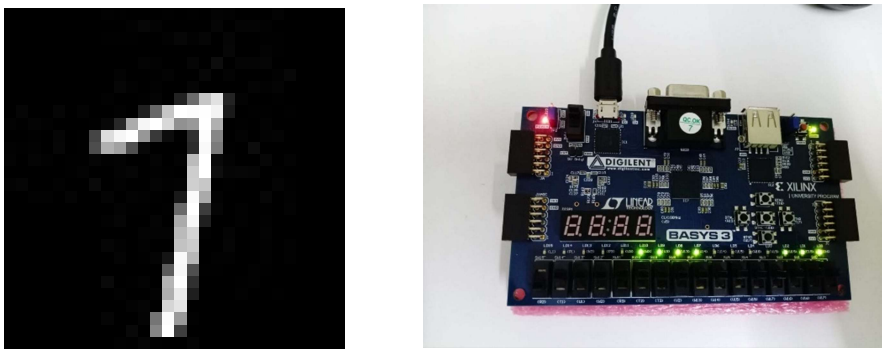

Fig 5.19 Output of the input test image5 on Basys3

5.6.1 Power report

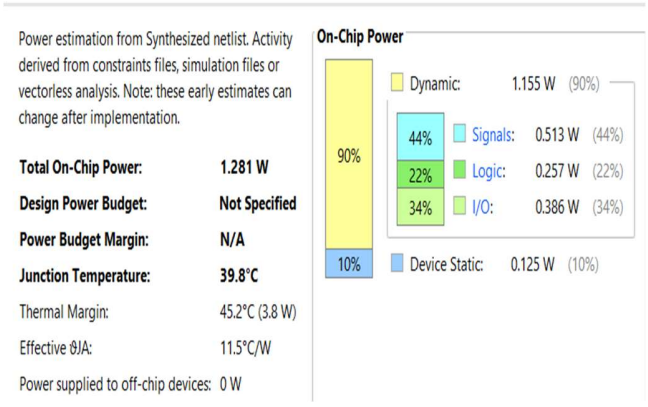
Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.
Total On-Chip Power: 1.281 W
Design Power Budget: Not Specified
Power Budget Margin: N/A
Junction Temperature: 39.8°C
Thermal Margin: 45.2°C (3.8 W)
Effective ΘJA: 11.5°C/W
Power supplied to off-chip devices: 0 W
On-Chip Power
90%
10%
Dynamic: 1.155 W (90%)
44% Signals: 0.513 W (44%)
22% Logic: 0.257 W (22%)
34% I/O: 0.386 W (34%)
Device Static: 0.125 W (10%)

Fig 5.20 Power report of image classification

6

CONCLUSION AND FUTURE SCOPE

6.1 Conclusion

The architecture of Spiking Neural Networks provided in this thesis achieved the low power consumption of 0.28W. This system can be used for the applications like image classification which consumes 1.28W power for the above classification.

Enhanced performance:

Demonstrated architecture has improved in processing speed and computational efficiency. Because of parallelism faster data processing and reduced latency.

Efficient resource utilization:

Optimal utilization of FPGA resources balancing the trade-off between performance and resource consumption

Scalability and flexibility:

Ease for scalability and flexibility enabling customization and adaptable to wider applications.

It works at 100 MHz Clock frequency. With this frequency of operation the power consumption is reduced.

6.2 Future Scope

. This architecture can be used for simple implementations like text and image recognition in remote areas where power and GPUs are limited resources in future. The availability of limited logic blocks and memory restricts the complexity and extent of networks that can be implemented on FPGA. Power consumption can also remain high for large networks to be implemented.

For large and complex networks like LLMs, ASIC is the best option as it consume low power and also low manufacturing costs in large scale. One can move to NoC (Network on Chip) architecture for the ASIC design of SNN. As Analog circuits are Process, Voltage and Temperature Variant (PVT), digital implementation is the best choice.

REFERENCES

[1] M. Capra, B. Bussolino, A. Marchisio, G. Masera, M. Martina and M. Shafique, "Hardware and Software Optimizations for Accelerating Deep Neural Networks: Survey of Current Trends, Challenges, and the Road Ahead," in *IEEE Access*, vol. 8, pp. 225134-225180, 2020, doi: 10.1109/ACCESS.2020.3039858.

[2] S. Kim, S. Kim, S. Um, S. Kim, J. Lee and H. -J. Yoo, "SNPU: An Energy-Efficient Spike Domain Deep-Neural-Network Processor With Two-Step Spike Encoding and Shift-and-Accumulation Unit," in *IEEE Journal of Solid-State Circuits*, vol. 58, no. 10, pp. 2812-2825, Oct. 2023, doi: 10.1109/JSSC.2023.3270442.

[3] Y. Liu, Y. Chen, W. Ye and Y. Gui, "FPGA-NHAP: A General FPGA-Based Neuromorphic Hardware Acceleration Platform With High Speed and Low Power," in *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 69, no. 6, pp. 2553-2566, June 2022, doi: 10.1109/TCSI.2022.3160693.

[4] X. Cheng, S. Cao, S. Wang, X. Zeng, W. Li and M. Wang, "A 1024-Neuron 1M-Synapse Event-Driven SNN Accelerator for DVS Applications," *2024 IEEE International Symposium on Circuits and Systems (ISCAS)*, Singapore, Singapore, 2024, pp. 1-5, doi: 10.1109/ISCAS58744.2024.10558416.

[5] Y. Kuang *et al*., "ESSA: Design of a Programmable Efficient Sparse Spiking Neural Network Accelerator," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 30, no. 11, pp. 1631-1641, Nov. 2022, doi: 10.1109/TVLSI.2022.3183126.

[6] W. Ye, Y. Chen and Y. Liu, "The Implementation and Optimization of Neuromorphic Hardware for Supporting Spiking Neural Networks With MLP and CNN Topologies," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, no. 2, pp. 448-461, Feb. 2023, doi: 10.1109/TCAD.2022.3179246.

[7] S. Li, Z. Zhang, R. Mao, J. Xiao, L. Chang and J. Zhou, "A Fast and Energy-Efficient SNN Processor With Adaptive Clock/Event-Driven Computation Scheme and Online Learning," in *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 4, pp. 1543-1552, April 2021, doi: 10.1109/TCSI.2021.3052885.

[8] J. Han, Z. Li, W. Zheng and Y. Zhang, "Hardware implementation of spiking neural networks on FPGA," in *Tsinghua Science and Technology*, vol. 25, no. 4, pp. 479-486, Aug. 2020, doi: 10.26599/TST.2019.9010019.

[9] J. Li, G. Shen, D. Zhao, Q. Zhang and Y. Zeng, "FireFly v2: Advancing Hardware Support for High-Performance Spiking Neural Network With a Spatiotemporal FPGA Accelerator," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 43, no. 9, pp. 2647-2660, Sept. 2024, doi: 10.1109/TCAD.2024.3380550.

[10] M. Heidarpur, A. Ahmadi, M. Ahmadi and M. Rahimi Azghadi, "CORDIC-SNN: On-FPGA STDP Learning With Izhikevich Neurons," in *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 7, pp. 2651-2661, July 2019, doi: 10.1109/TCSI.2019.2899356.

[11] H. Chu *et al.*, "A Neuromorphic Processing System With Spike-Driven SNN Processor for Wearable ECG Classification," in *IEEE Transactions on Biomedical Circuits and Systems*, vol. 16, no. 4, pp. 511-523, Aug. 2022, doi: 10.1109/TBCAS.2022.3189364.

[12] Z. Cao, Z. Zhang, Q. Sun, B. Hou, L. Jiao and Y. Yang, "Cost-Effective and High-Speed Implementation of Brain-Like Spiking Neural Network with Encoding Architectures on FPGA," in *CHAIN*, vol. 1, no. 2, pp. 167-176, June 2024, doi: 10.23919/CHAIN.2024.000003.

[13] J. Sommer, M. A. Özkan, O. Keszocze and J. Teich, "Efficient Hardware Acceleration of Sparsely Active Convolutional Spiking Neural Networks," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 11, pp. 3767-3778, Nov. 2022, doi: 10.1109/TCAD.2022.3197512.

[14] H. Chen *et al.*, "Research on Hardware Acceleration of Traffic Sign Recognition Based on Spiking Neural Network and FPGA Platform," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 33, no. 2, pp. 499-511, Feb. 2025, doi: 10.1109/TVLSI.2024.3470834.

[15] M. T. L. Aung *et al.*, "DeepFire2: A Convolutional Spiking Neural Network Accelerator on FPGAs," in *IEEE Transactions on Computers*, vol. 72, no. 10, pp. 2847-2857, Oct. 2023, doi: 10.1109/TC.2023.3272284.

[16] Y. Chen, W. Ye, Y. Liu and H. Zhou, "SiBrain: A Sparse Spatio-Temporal Parallel Neuromorphic Architecture for Accelerating Spiking Convolution Neural Networks With Low Latency," in *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 71, no. 12, pp. 6482-6494, Dec. 2024, doi: 10.1109/TCSI.2024.3393233.

[17] W. Liu, S. Xiao, Y. Liu and Z. Yu, "SC-PLR: An Approximate Spiking Neural Network Accelerator With On-Chip Predictive Learning Rule," in *IEEE Transactions on Biomedical Circuits and Systems*, vol. 18, no. 5, pp. 1156-1165, Oct. 2024, doi: 10.1109/TBCAS.2024.3385235.

[18] G. Chen *et al.*, "An Edge Neuromorphic Processor With High-Accuracy On-Chip Aggregate-Label Learning," in *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 72, no. 3, pp. 509-513, March 2025, doi: 10.1109/TCSII.2025.3529670.

[19] B. Deng, Y. Fan, J. Wang and S. Yang, "Reconstruction of a Fully Paralleled Auditory Spiking Neural Network and FPGA Implementation," in IEEE Transactions on Biomedical Circuits and Systems, vol. 15, no. 6, pp. 1320-1331, Dec. 2021, doi: 10.1109/TBCAS.2021.3122549.

[20] X. Wang, C. Feng, X. Kang, Q. Wang, Y. Huang and T. T. Ye, "RV-SCNN: A RISC-V Processor With Customized Instruction Set for SNN and CNN Inference Acceleration on Edge Platforms," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 44, no. 4, pp. 1567-1580, April 2025, doi: 10.1109/TCAD.2024.3472293.

[21] Jaiswal, V., T, N.B., Murugan, P. *et al.* Fault analysis on deep groove ball bearing using ResNet50 and AlexNet50 algorithms. *Sci Rep* **15**, 12962 (2025). https://doi.org/10.1038/s41598-025-97410-8

[22] Tengxiao Wang, Cong Shi, Xichuan Zhou, Yingcheng Lin, Junxian He, Ping Gan, Ping Li, Ying Wang, Liyuan Liu, Nanjian Wu, Gang Luo,"CompSNN: A lightweight spiking neural network based on spatiotemporally compressive spike features, Neurocomputing, Volume425,2021, Pages 96-106, ISSN0925-2312,

https://doi.org/10.1016/j.neucom.2020.10.100.(https://www.sciencedirect.com/science/article/pii/S0925231220317136)

[23] S. Li, Z. Zhang, R. Mao, J. Xiao, L. Chang and J. Zhou, "A Fast and Energy-Efficient SNN Processor With Adaptive Clock/Event-Driven Computation Scheme and Online Learning," in *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 4, pp. 1543-1552, April 2021, doi: 10.1109/TCSI.2021.3052885.

[24] Guo, Wenzhe & Yantır, Hasan & Fouda, Mohammed E. & Eltawil, Ahmed & Salama, Khaled. (2020). Towards Efficient Neuromorphic Hardware: Unsupervised Adaptive Neuron Pruning. Electronics. 9. 1059. 10.3390/electronics9071059.

[25] Lee, Chankyu & Sarwar, Syed & Panda, Priyadarshini & Srinivasan, Gopalakrishnan & Roy, Kaushik. (2020). Enabling Spike-Based Backpropagation for Training Deep Neural Network Architectures. Frontiers in Neuroscience. 14. 119. 10.3389/fnins.2020.00119.

[26] R. Toole *et al*., "Photonic Implementation of Spike-Timing-Dependent Plasticity and Learning Algorithms of Biological Neural Systems," in *Journal of Lightwave Technology*, vol. 34, no. 2, pp. 470-476, 15 Jan.15, 2016, doi: 10.1109/JLT.2015.2475275.

[27] C. Chen, Y. Dai and H. Hong, "A Neuromorphic Spiking Neural Network Using Time-to-First-Spike Coding Scheme and Analog Computing in Low-Leakage 8T SRAM" in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 32, no. 05, pp. 848-859, May 2024, doi: 10.1109/TVLSI.2024.3368849.

[28] https://ecrypt-eu.blogspot.com/2018/02/fast-fully-homomorphic-evaluation-of.html

[29] https://digilent.com/reference/programmablelogic/basys3/start?srsltid=AfmBOopL9t_WDq1YEDmPN6208eNr0kFToFmksna51bGxwZwnnxurmhSd

[30] https://www.ni.com/en/shop/compactrio/what-are-compactrio-controllers/advantages-of-xilinx-7-series-fpga-and-soc-devices.html